DISTRIBUTED CONTROL APPROACHES TO NETWORK OPTIMIZATION

A Thesis

by

SANKALP SAH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2010

Major Subject: Computer Engineering

DISTRIBUTED CONTROL APPROACHES TO NETWORK OPTIMIZATION

A Thesis

by

SANKALP SAH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Srinivas Shakkottai
Committee Members,	Narasimha Reddy
	Natarajan Gautam
Head of Department,	Costas N. Georghiades

May 2010

Major Subject: Computer Engineering

ABSTRACT

Distributed Control Approaches to Network Optimization. (May 2010)

Sankalp Sah, B.Tech., Indian Institute of Technology Guwahati

Chair of Advisory Committee: Srinivas Shakkottai

The objective of this research is to develop distributed approaches to optimizing network traffic. Two problems are studied, which include exploiting social networks in routing packets (coupons) to desired network nodes (users in the social network), and developing a rate based transport protocol, which will guarantee that all the flows in a network (e.g. Internet) meet a delay constraint per packet.

Firstly, we will study social networks as a means of obtaining information about a system. They are increasingly seen as a means of obtaining awareness of user preferences. Such awareness could be used to target goods and services at them. We consider a general user model, wherein users could buy different numbers of goods at a marked and at a discounted price. Our first objective is to learn which users would be interested in a particular good. Second, we would like to know how much to discount these users such that the entire demand is realized, but not so much that profits are decreased. We develop algorithms for multihop forwarding of such discount coupons over an online social network, in which users forward coupons to each other in return for a reward. Coupling this idea with the implicit learning associated with backpressure routing (originally developed for multihop wireless networks), we would like to demonstrate how to realize optimal revenue. We will then propose a simpler heuristic algorithm and try to show, using simulations, that its performance approaches that of backpressure routing.

As the second problem, we look at the traditional formulation of the total value of information transfer, which is a multi-commodity flow problem. Here, each data

source is seen as generating a commodity along a fixed route, and the objective is to maximize the total system throughput under some concept of fairness, subject to capacity constraints of the links used. This problem is well studied under the framework of network utility maximization and has led to several different distributed congestion control schemes. However, this idea of value does not capture the fact that flows might associate value, not just with throughput, but with link-quality metrics such as packet delay, jitter and so on. The traditional congestion control problem is redefined to include individual source preferences. It is assumed that degradation in link quality seen by a flow adds up on the links it traverses, and the total utility is maximized in such a way that the quality degradation seen by each source is bounded by a value that it declares. Decoupling source-dissatisfaction and link-degradation through an "effective capacity" variable, a distributed and provably optimal resource allocation algorithm is designed, to maximize system utility subject to these quality constraints. The applicability of our controller in different situations is illustrated, and results are supported through numerical examples.

To My Parents & Sai Baba

ACKNOWLEDGMENTS

It takes a lot to pursue a goal in life. For this thesis, I have every reason to be thankful to my advisor, Dr. Srinivas Shakkottai. Formulating such interesting problems for the research work of a master's student is one thing that goes a long way towards completion of the thesis. His continued motivation for me by being a role model has kept me in focus for the task at hand. Every student should be given the kind of freedom I received while working on my thesis. Thank you Dr. Shakkottai!

I also take this opportunity to thank Dr. Narasimha Reddy and Dr. Natarajan Gautam for being willing to be on my committee.

This also gives me a chance to pay my gratitude to my parents without whom I would not be the engineer I am today. It is their sacrifices that have made me the person I am. The constant support by my brother to keep on going has been valuable and I thank him for that.

Thank you Tammy and Carolyn for helping me out during my entire Master's Degree; both of you have been very helpful.

Last, but not least, I thank the Almighty God for giving me strength to overcome my limitations all the time.

TABLE OF CONTENTS

CHAPTER Page

I INTRODUCTION 1

A. Implicit Learning for Explicit Discount Targeting in Online Social Networks 1
B. Value-aware Resource Allocation for Service Guarantees in Networks 5

II IMPLICIT LEARNING FOR EXPLICIT DISCOUNT TARGETING IN ONLINE SOCIAL NETWORKS 9

A. Related Work 9
B. System Model 10
C. Profit Maximization 12
D. Coupon Distribution 13
1. Small Time Scale Control: Backpressure Coupon Routing 14
2. Large Time Scale Control: Coupon Rate Selection . . 19
E. Delay-Based Coupon Forwarding 21
F. Simulation Results 23
1. Simple Tree Topology 23
2. Power Law Topology 27

III VALUE-AWARE RESOURCE ALLOCATION FOR SERVICE GUARANTEES IN NETWORKS 30

A. Main Results 30
B. Examples of Quality Degradation Functions 31
1. Average Delay in M/M/1 Queues 32
2. Decay-rate of a Fluid Buffer With On-Off Service Process 32
C. Centralized Resource Allocation 33
1. Homogeneous Tandem Network 34
2. Three-Flows Two-Hop Network 37
D. Primal Algorithm 38
E. Dual Algorithm 40
1. Distributed Algorithm 42

CHAPTER Page

2. Global Stability of Distributed Algorithm 45
3. Homogeneous Tandem Network 46
4. Three-Flows Two-Hop Network 48
F. NS-2 Experiments 49
1. Source Dynamics 49
2. Link (Router) Dynamics 50
3. Results 53
a. Three-Flows Two-Hop Network 53
b. Abilene Topology 58
c. Abilene Topology: ON-OFF flows 62
d. Access-Core Topology 64

IV CONCLUSIONS 69

A. Implicit Learning for Explicit Discount Targeting in Online Social Networks 69
B. Value-aware Resource Allocation for Service Guarantees in Networks 69

REFERENCES 71

APPENDIX A 75

APPENDIX B 76

APPENDIX C 79

VITA 80

LIST OF FIGURES

FIGURE		Page
1	A coupon distribution system	10
2	Example trajectories of fractional errors, random distribution	24
3	Example trajectories of fractional errors, backpressure distribution .	25
4	Example trajectories of fractional errors, delay distribution	25
5	Simple tree topology.	26
6	An example trajectory for user 3 over the small time scale. The solid line indicates purchases made at the marked price, while the dashed line indicates discounted purchases. The user has $\hat{x}_j^l = 50$ and $\hat{x}_j^h = 60$. In this example we have set (for illustration) $\gamma_j = \hat{x}_j^l = 50$, i.e., the reward rate for coupon forwarding is known exactly, and we see that the user receives exactly the right number of coupons.	27
7	Trajectory of revenue obtained by the store using different schemes.	28
8	Trajectory of revenue obtained by the store using different schemes for the power-law topology.	29
9	Tandem network of L links being shared by n flows.	35
10	Three flows sharing a two-link network.	37
11	A block diagram of value-aware resource allocation that allows decoupling of user-dissatisfaction on the source side, and quality on the link side.	44
12	Distributed resource allocation for homogeneous tandem network. . .	47
13	Distributed resource allocation for three-flow two-hop network. . . .	49

FIGURE Page

14 Link in our system comprises of the TS-queue and the router queue. 51

15 TS-queue in our system 51

16 Router queue in our system 52

17 Three flows sharing a two-link network. 53

18 Rates for the routes 1, 2 and 3 for a realizable but, tight constraint . 55

19 Delay experienced by route 3, as plotted against the required delay and the control delay 55

20 $\tilde{y}$ at router node 1 56

21 Rates for the routes 1, 2 and 3 for a realizable loose constraint 57

22 $\tilde{y}$ at router node 1 57

23 Abeline topology 58

24 Rates for the routes 1, 2 and 3 for a realizable but, tight constraint . 59

25 Delay experienced by route 1, as plotted against the required delay and the control delay 60

26 $\tilde{y}$ at router node 10 60

27 Rates for the routes 1, 2 and 3 for a realizable loose constraint 61

28 $\tilde{y}$ at router node 10 62

29 Rates for flows x_1, x_2, x_3 63

30 $\tilde{y}$ at router node 10 63

31 Access core topology 64

32 Rates for the routes 1, 2 and 3 for a realizable but, tight constraint . 65

33 Delay experienced by route 1, as plotted against the required delay and the control delay 66

FIGURE Page

34 $\tilde{y}$ at router node 10 66

35 Rates for the routes 1, 2 and 3 for a realizable loose constraint 67

36 $\tilde{y}$ at router node 10 68

CHAPTER I

INTRODUCTION

A. Implicit Learning for Explicit Discount Targeting in Online Social Networks

The past few years have seen the rapid and global emergence of online social networks as a medium for community interaction [1, 2]. Their success can be gauged by their meteoric adoption by a large populace, and the continued success of the medium requires a sound economic foundation for sustainable growth. The medium of choice to extract commercial value out of online social networks is the advertising and sale of goods and services by using the structure and nature of social interactions. Since an individual user's preferences can be identified by his or her response to exogenous stimulation such as advertisements, an approach that is often used is to try to learn about user preferences through exploration. Such feedback obtained though the user responses could be used to offer incentives to purchase certain goods and services.

Consider goods and services that are consumed periodically (say on a weekly or monthly basis) such as movie tickets, car washes, fitness club visits and so on. Here, we could have a high displayed price that some consumers would be willing to pay. In order to extract maximum revenue, other consumers need to be subsidized to some extent by using discounts such as rebate coupons. In other words, discount coupons are used to create multiple tiers of prices for the same good or service. Two questions immediately arise, (i) which users should be given coupons?, and (ii) how many coupons should they be given? Further, the questions have to be answered in a system in which user preferences change over time.

Both questions are hard since the seller of the good is unlikely to be aware of the

---

The journal model is *IEEE Transactions on Automatic Control*.

preferences of users, or possibly even of their existence. Even if the seller is aware of a user's interest, he must not give too many or too few discounts – too many would reduce profits and too few would mean that the entire demand would not be realized. There are two classical methods of offering such incentives. The first is to flood communities of users in the hope that some of them would use the coupons. Here, the idea is to pre-identify communities that are not likely to buy the good without discounts. If identification is incorrect, either the users would not use the coupons, or the wrong set of users would be discounted. The second is to rely on self-identification of interested individuals. Here, the store gets the users to sign up for coupons, and then judiciously sends them coupons. Such a scheme would work only on users who do identify themselves to the store, and might not realize the entire demand.

Both the above solutions ignore the fact that users could belong to an online social network, and hence could obtain coupons by interacting with his or her friends. Thus, users could forward coupons from one to the next in a multihop fashion across the online social network. If a user is interested in good that the coupon represents, he or she could use it. Otherwise the user could forward it onwards. Allowing for coupon forwarding implies that the two questions raised have to be modified slightly: (i) given that a user has a coupon that he does not want, which friend should he forward it to and why?, and (ii) what rate should coupons be injected into the system? Hence, we need to design a signaling scheme that incentivizes users to somehow learn the preferences of users in such a way that the profits of the store are maximized. An example of such a system in practice is *mGinger* [3] that acts as a multi-hop advertising and discount distribution system using SMS messages, with rewards being paid in a pyramidal fashion. The motivation for multi-hop coupon distribution is that since user preferences change with time, and new products are continuously introduced, it is impossible for any store to be aware of all its potential

customers. Hence a system must *learn* user preferences, which then change after a while.

In this research, we develop *implicit distributed learning* schemes based on ideas of backpressure [4] that has been used as a throughput optimal scheme for packet routing in multihop wireless networks [5–8]. We assume that the capacity for consumption of a good i by user j can be divided naturally into two parts—one at at the marked ("high") price $\hat{x}_j^{ih}$, and one at the discounted ("low") price $\hat{x}_j^{il}$. We assume that these values are fixed for some duration of time, and so can be learned. The number of coupons given to the user must be carefully regulated; if it is larger than $\hat{x}_j^{il}$, the store loses profits due to excessive discounting, while if it is less than $\hat{x}_j^{il}$, the entire demand is not realized. We combine ideas of self-identification by users and directed flooding through backpressure to achieve an optimal solution that realizes the entire demand by injecting the right number of coupons, and maximizes profit by ensuring that the users receive coupons at the optimal rate.

We then use *optimization decomposition* techniques in Section D to develop a coupon distribution scheme consisting of three entities: (i) a store at which goods may be purchased, (ii) users connected by an online social network and (iii) a coupon source (or sources). The behavior of these entities is as follows:

- A store sells goods i at a marked price p^i, which it discounts to a price q^i upon presentation of a coupon. The store assigns a "goodness value" to each user j that makes a purchase form it. This value determines the probability with which neighbors of j are rewarded for forwarding a coupon to j[1]. However, all the other users (non-neighbors of j) that are involved in the forwarding path are

[1]Throughout this report, we use the word *reward* to denote remuneration for *forwarding* coupons, and the word *discount* to denote remuneration for redeeming coupons (when purchasing goods) at a store.

guaranteed a reward. This artifice enables locality of information, as we show later. In other words, although the discount carried by each coupon is identical, the reward for forwarding coupons to each user j is not. The store uses a simple *up-down controller* to determine the reward probability for forwarding, based on the number of goods purchased by user j.

- All users in the system maintain a count of the number of coupons of each kind that their neighbors possess via communication over the social network. Users also maintain a count of the number of unrewarded coupons associated with their neighbors by polling the relevant store. We refer to the sum of these two as the *effective coupons*. Coupons can be transferred among users in a multihop fashion, and users are incentivized to forward coupons in the direction of lowest *effective pressure*, i.e., to a neighbor who has the smallest number of effective coupons. This controller is similar in nature to a backpressure controller.

- Finally, the coupon source generates coupons of different kinds (corresponding to different goods), and sends them to users that have identified themselves as interested in receiving particular kinds of coupons. The source chooses to send coupons using a *threshold controller*, which generates new coupons when the effective pressure is less than a certain threshold.

We prove that the system using this *backpressure-based* coupon distribution evolves with time to attain the maximum profits by ensuring that each potential consumer obtains exactly the right number of coupons. The system is distributed and each user only requires information associated with his or her neighbors. Thus, it succeeds in achieving light-weight learning framework, in which exploring for user capacity and exploiting existing capacity go hand-in-hand.

We then consider a simpler heuristic algorithm in Section E, that is based on the

delay in obtaining rewards. This *delay-based* algorithm does not require information exchange between users. At any time, users simply forward coupons to that neighbor for whom the delay experienced between forwarding a coupon and obtaining a reward for that coupon is the smallest. This algorithm inherently captures the idea of backpressure, although it is at a coarse level.

Our final scheme is even simpler, and consists of *random* coupon distribution. Here, each user randomly forwards coupons to its neighbors in the hope of finding correct paths. This system does not learn user preferences. We use this algorithm to test the efficacy of our other algorithms.

We simulate the distribution schemes in Section F on different topologies to compare their performance. We show that the backpressure-based scheme achieves near-optimal revenue, while the delay-based scheme performs acceptably well. Further, both schemes significantly outperform the randomized scheme, thus making a strong case for backpressure based targeted coupon delivery in online social networks.

B. Value-aware Resource Allocation for Service Guarantees in Networks

Recent years have seen an enormous growth in demand for Internet access, with applications ranging from personal use to commercial and military operations. Several of these applications are sensitive to a "quality" of packet delivery. For instance, although archiving data transfer can tolerate long delays, voice over Internet protocol (VoIP) is very sensitive to latency. Between these two extreme examples lies a spectrum of applications with varying service requirements, e.g. electronic commerce, video conferencing and online gaming. All these applications require the allocation of enough network resources for satisfactory performance.

The design of efficient network control systems demands that end-user value be

taken into consideration when allocating resources. The Internet architecture is built around the concept of a *flow*, which is a transfer of data between a fixed source-destination pair. How do we quantify the value of such a flow? The classical formulation of the total value of information transfer is a multi-commodity flow problem, in which each data source is seen as generating a commodity along a fixed route, and the objective is to maximize the total throughput under some concept of fairness, subject to capacity constraints of the links used [15–18]. If the flow from source r has a rate $x_r \geq 0$ and the system utility associated with such a flow is represented by a concave, increasing function $U_r(x_r)$, the objective is

$$\begin{aligned} \max \quad & \sum_{r \in \mathcal{S}} U_r(x_r) \\ \text{s.t.} \quad & y_l \leq c_l, \ \forall \ l \in \mathcal{L} \end{aligned} \tag{1.1}$$

where $\mathcal{S}$ is the set of sources, $\mathcal{L}$ the set of links, c_l the capacity of link $l \in \mathcal{L}$. Also let R be the routing matrix with $R_{lr} = 1$ if the route associated with source r uses link l. The load on link l is $y_l = \sum_{r \in \mathcal{S}} R_{lr} x_r$. Note that we refer to flows and sources interchangeably; if there are multiple flows between a source and a destination, we simply give them different names. This is a convex optimization problem that is well studied [15–18] under the framework of network utility maximization.

This approach to network resource allocation often can be used to decompose the problem into several sub problems, each of which are amenable to distributed solution. This so-called *optimization decomposition framework* has yielded a rich set of control schemes and protocols, whose architectural implications are discussed in detail in [19]. For example, there is a strong connection between the so-called *primal solution* to the utility maximization problem and TCP-Reno [20, 21] characterized in [22, 23]. Similarly, one can obtain connections between TCP-Vegas and the *dual solution* of the

problem [24]. The same approach has been taken in the design of several new protocols such as Scalable TCP [25,26] (that allows scaling of rate increases/decreases based on network characteristics), FAST-TCP [27] (meant for high bandwidth environments), TCP-Illinois [28] (that uses loss and delay signals to attain high throughput), and TRUMP [29] (a multipath protocol with fast convergence properties).

However, there is a growing realization that throughput cannot be considered as the sole value metric. As mentioned above, in applications such as voice calls, the data is rendered useless after a certain delay threshold. Thus, simply ensuring that the link capacity is not exceeded is not sufficient to provide value in this scenario – how do we ensure that the user is not dissatisfied with the quality of service? In many cases the quality of data transfer over a link decreases with load. For example, metrics such as the delay and the jitter experienced by packets as they pass through a queue depend on the total load on the link. Such quality degradation might also add up over multiple hops. Indeed, the delay experienced by packets in a flow is the sum of the delays experienced over each hop taken.

Once we have a clear conception of quality degradation as a function of link load, we ask the following question: *can we design a simple distributed algorithm for fair resource allocation under which each users' quality is no worse than a value that she or he declares?* We need to redefine the traditional congestion control problem to include individual source preferences. We denote the degradation in quality of link l with load y_l by a convex increasing function $V_l(y_l)$, and assume that degradation in link quality seen by a flow adds up on the links it traverses. In addition, we assume that the quality degradation is inherent to a link, and is identical for all flows sharing the link. Thus, there are no priorities for any particular flows. We now maximize utility in such a way that the total degradation seen by each source r is required to

be bounded by a value σ_r. Thus, the modified objective is

$$\begin{aligned} &\max_{x_r \geq 0, y = Rx} \sum_{r \in \mathcal{S}} U_r(x_r) \\ &\text{subject to } \sum_{l \in \mathcal{L}} R_{lr} V_l\left(y_l\right) \leq \sigma_r, \end{aligned} \tag{1.2}$$

where we assume that $\lim_{y \to c_l} V_l(y) = \infty$. Note that this is a convex optimization problem where the quality degradation on each route is bounded. While some of our objectives might be achieved by existing schemes such as DiffServ [30], they often require complex priority management methods and per-flow information to be maintained at routers. In this work, our objective is to design a simple distributed control scheme that can achieve this goal without maintaining per-flow information or prioritizing certain packets at intermediate hops. We overview our main contributions below, with details in the sections following.

CHAPTER II

IMPLICIT LEARNING FOR EXPLICIT DISCOUNT TARGETING IN ONLINE SOCIAL NETWORKS

A. Related Work

A typical learning problem is that of the *bandit-problem* [9–11], which has its historical origin in a coin-operated gambling machine that pays off according to the matching of symbols on wheels spun by a handle, also called a one-armed bandit. The multi-armed bandit is the situation confronted by a gambler who has a choice between n one-armed bandits, and attempts to learn by experience which one to pull. Our approach to learning is quite different, and follows a more implicit method derived from communication networks.

The backpressure algorithm is a joint routing/scheduling for communication networks where the routing/scheduling decisions are dynamically made without requiring the information of the network topology and traffic arrivals [4]. The algorithm has been proved to stabilize any traffic that can be stabilized by any other rotuing/scheduling algorithm for different types of communication networks [5–8, 12, 13].

The backpressure algorithm learns traffic and network information implicitly from queue-lengths. Based on the similar idea, we develop a coupon delivery scheme, where the users learn the coupon demands from coupon queues and dynamically distribute coupons to their neighbors based on the effective pressure. However, coupon delivery problem has fundamental differences from routing/scheduling for communication networks, which include: (i) user behavior follows user beliefs, so a proper incentive scheme is needed to motivate users to distribute coupons in the right way; (ii) users are selfish and will use all the coupons they get, so a good coupon delivery

scheme should avoid sending too many coupons to a user. This implies that a user that is interested in a particular type of coupons cannot serve as a relay for that type of coupon. This is fundamentally different from communication networks where every node can serve as a relay.

B. System Model

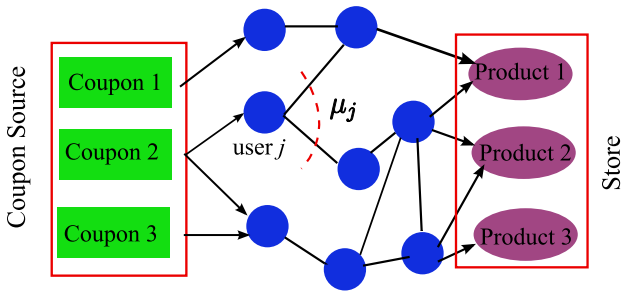

Fig. 1. A coupon distribution system

Network model: We consider an online social network structure as shown in Figure 1. Denote by $\mathcal{N}$ the set of nodes and $\mathcal{L}$ the set of links. There are three different nodes — coupon distributor, users, and store — in the network. The links represent social connections. A link from a coupon source to a user represents the idea that the user has registered with the source to receive its coupon periodically. A link from a user to a product means that the user buys that product periodically. The links between users are assumed to be bidirectional, and represent friendship between the connected users. In this research, we assume that the coupon sources and the store are managed by the same entity. We use s to denote the store and d to denote the coupon distributor. We define $\mathcal{S}$ to be the set of products and $\mathcal{B}_i$ is the set of users who will buy product i.

We consider a synchronized slotted-time system. We define μ_j to be the coupon

transmission capacity of node j, which is the maximum number of coupons user j can send out in one time-step. We also impose the constraint that a user can buy a discounted product only if a coupon is presented.

Two-capacity model for user demands: We assume that users naturally have a maximum number of goods that they would buy at the marked price, $\hat{x}_j^{ih}$, and number of goods that users would buy at the discounted price, $\hat{x}_j^{il}$. Note that either of these quantities could be zero. We further define $b_j^i = \hat{x}_j^{ih} + \hat{x}_j^{il}$. These values can be thought of as the *capacities* associated with a user. We consider two different time scales in this research. The small time scale t is the one in which purchases are made and coupons are delivered. The large time-scale, consisting of T small time slots, is the buying interval after which the customers start afresh. Since the users have the incentive to buy a product with a low price, we assume that the $\hat{x}_j^{ih}$, associated with the high price goods could be used to buy low priced goods as well. Specifically, during each large time scale, if the user were given no more than $b_j^i T$ coupons, he would use them all and buy $b_j^i T$ goods.

Note that if a user were given more than $\hat{x}_j^{il} T$ coupons, the store would not extract the maximum extractable revenue. If he were given less than $\hat{x}_j^{il} T$ coupons, he would not buy enough discounted goods, which reduces the profit of the store as well. A store that is unaware of these two capacities needs to probe customers in order to find their true potential, and neither supply too few or too many coupons. In what follows, we present a distributed solution that automatically explores for and attains the capacity of users, thus achieving the profit maximizing solution.

C. Profit Maximization

Consider the profit made by the store. We say a coupon is *valid* if it is eventually used to purchase a product. We denote by $y^i_{(m,n)}$ the average number of valid type-i coupons sent from user m to user n in a time slot. The profit the store extracts from user j is

$$q^i y^i_{(j,s)} + p^i \min\left\{\hat{x}^{ih}_j, b^i_j - y^i_{(j,s)}\right\}.$$

Thus, the maximum profit the store can extract is defined by the following optimization problem:

OPT 1:

$$\max \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{B}_i} \left(q^i y^i_{(j,s)} + p^i \min\left\{\hat{x}^{ih}_j, b^i_j - y^i_{(j,s)}\right\}\right) \tag{2.1}$$

$$s.t. \quad \sum_{i\in\mathcal{S}} \sum_{j:(m,j)\in\mathcal{L}, j\neq s_i} y^i_{(m,j)} \leq \mu_m, \forall m \in \mathcal{N} \tag{2.2}$$

$$\sum_{j:(m,j)\in\mathcal{L}} y^i_{(m,j)} = \sum_{n:(n,m)\in\mathcal{L}} y^i_{(n,m)} \forall m \in \mathcal{N} \tag{2.3}$$

$$y^i_{(m,j)} = 0 \text{ if } m \in \mathcal{B}_i \text{ and } j \neq s \tag{2.4}$$

where (2.2) is the capacity constraint, which indicates node m cannot send more than μ_m coupons in a time-slot, (2.3) is the flow-conservation constraint for the coupons, and (2.4) indicates that user j will not forward type-i coupons to his/her neighbors if he/she uses type-i coupons.

To extract the maximum revenue, we need to distribute coupons to the users. There are two difficulties in distributing the right number of coupons to the users:

(1) The optimal $\left(\hat{x}^{ih}_j, \hat{x}^{il}_j\right)$ is unknown at the store, and needs to be identified.

(2) Since all users interested in a product may not be registered to directly receive coupons, they might need to receive such coupons via the social network. The

store cannot directly control the number of coupons sent to such users.

To tackle these two difficulties, we develop a two time-scale coupon distribution scheme in the next section.

D. Coupon Distribution

In this section, we develop an implicit distributed learning scheme based the idea of backpressure routing/scheduling [4]. Our algorithm consists of two control loops that operate at the small time scale and the large time scale. The purpose of the control loops is as follows:

1. **Choice of Coupon Forwarding Reward Rate:** At the large time scale, each store must adapt the target rate γ_j^i for the next buying interval using the information gathered about the customers' preferences over the past intervals. In our algorithm, γ_j^i is an estimate of $\hat{x}_j^{il}$. As discussed in Section B, if γ_j^i is set too low, customer j may not purchase all the goods that he potentially could, and if γ_j^i is too high, customer j may be being discounted excessively and the store is not extracting the maximum extractable revenue.

2. **Coupon Routing at Target Rate:** At the small time scale a store arbitrarily assigns to a rate of coupon delivery γ_j^i to each product i and each customer j that purchases goods from it. The purpose of this control loop is to ensure that the customer would indeed receive discount coupons at this target rate. Mathematically, we will guarantee that the coupon distribution algorithm solves the following optimization problem:

   OPT 2:

$$\max \sum_{i \in \mathcal{S}} q^i \left(\sum_{j \in \mathcal{B}_i} y^i_{j,s_i} \right) \tag{2.5}$$

$$s.t. \quad \sum_{i \in \mathcal{S}} \sum_{j:(m,j) \in \mathcal{L}, j \neq s_i} y^i_{(m,j)} \leq \mu_m, \forall m \in \mathcal{N} \tag{2.6}$$

$$\sum_{j:(m,j) \in \mathcal{L}} y^i_{(m,j)} = \sum_{n:(n,m) \in \mathcal{L}} y^i_{(n,m)} \forall m \in \mathcal{N} \tag{2.7}$$

$$y^i_{(j,s)} \leq \gamma^i_j \quad \forall j, i \tag{2.8}$$

$$y^i_{(m,j)} = 0 \text{ if } m \in \mathcal{B}_i \text{ and } j \neq s \tag{2.9}$$

To show the correctness of the proposed algorithm, we first need the following straightforward lemma.

Lemma 1 *Given that $\gamma^i_j = \hat{x}^{il}_j$ for all i and j, OPT 1 is equivalent to OPT 2.*

Proof 1 *The proof is presented in Appendix A.*

Next, we develop a distributed coupon routing algorithm that solves OPT 2.

1. Small Time Scale Control: Backpressure Coupon Routing

We first introduce the coupon management scheme which consists of three parts: (i) each user maintains a per-product queue, and monitors the lengths of the queues; (ii) store rewards the neighbors that forwarded type i coupons used by each customer j at a rate γ^i_j, and monitors the number of unrewarded coupons at each customer[1]; (iii) coupon distributor i monitors the number of coupons she has not yet sent out, and generates additional coupons based on this value.

A1: Coupon Management:

[1]Recall that these are coupons that have been redeemed for a discount by j, but the neighbors of j who forwarded these coupons have not been rewarded.

(1) Per-product queues are maintained at each user, and the number of type i coupons user j has at a finer time-step t is denoted by $Q_j^i[t]$. Thus, the dynamic of $Q_j^i[t]$ is as follows: If $j \notin \mathcal{B}_i$, then

$$\begin{aligned} & Q_j^i[t+1] \\ = & \left(Q_j^i[t] + \sum_{m:(m,j)\in\mathcal{L}} y_{(m,j)}^i[t] - \sum_{n:(j,n)\in\mathcal{L}} y_{(j,n)}^i[t] \right)^+ ; \end{aligned}$$

otherwise

$$Q_j^i[t+1] = Q_j^i[t] + \sum_{m:(m,j)\in\mathcal{L}} y_{(m,j)}^i[t] - y_{(j,s)}^i[t],$$

where

$$\begin{aligned} y_{(j,s)}^i[t] \;\; = \;\; \min \Bigg\{ & Q_j^i[t] + \sum_{m:(m,j)\in\mathcal{L}} y_{(m,j)}^i[t], \\ & \left(b_j^i T - \sum_{\tau=0}^{t-1} y_{(j,s)}^i[\tau] \right)^+ \Bigg\}, \end{aligned}$$

i.e., user j will use up all available coupons unless she has already bought enough $(b_j^i T)$ products.

(2) Store maintains a queue for unrewarded coupons corresponding to each of product i and its customers j. We may think of these as *virtual coupons* that are used to maintain a pressure on j's neighbors. Note that it is only the *neighbors* of j that are not rewarded for forwarding these coupons, the rest of the users involved in forwarding coupons would be guaranteed a reward (and, of course, j has already redeemed these coupons for a discount). Denote by $\tilde{Q}_j^i[t]$ the number of such unrewarded coupons corresponding to customer j. We have

$$\tilde{Q}_j^i[t+1] = \left(\tilde{Q}_j^i[t] + y_{(j,s)}^i[t] - \gamma_j^i \right)^+,$$

where γ_j^i is the coupon forwarding reward rate for neighbors of customer j.

(3) Coupon distributor d maintains a separate queue for each type of coupons that have not been sent out. The length of the queue is denoted by $\tilde{Q}_d^i[t]$ for each product i. We have

$$Q_d^i[t+1] = \left(Q_d^i[t] + \Theta^i[t] - \sum_{j:(d_i,j)\in\mathcal{L}} y_{(d,j)}^i[t] \right)^+,$$

where $\Theta^i[t]$ is the number of new type i coupons generated by coupon distributor i at time t.

(4) We also assume that when user j receives a type i coupon such that $j \notin \mathcal{B}_i$, user j will insert her identity and the coupon queue length $Q_j^i[t]$ in the coupon before sending the coupon to her neighbor. This information allows the store to reconstruct path and reward the coupon relays based on $Q_j^i[t]$.

In our system the store need to reward coupon fowarding in order to motivate users to forward coupons to their friends. The efficiency of a coupon distribution scheme is determined by: (i) the incentive scheme that the store use, and (ii) the users' decisions under the incentive scheme. Next, we propose a coupon rewarding scheme, under which a rational user will distribute the coupons according to a backpressure policy. The optimality of the coupon distribution scheme will be proved in Theorem 2.

A2: Reward Scheme for Coupon Forwarding: Store rewards the users involved in forwarding each used type i coupon with a total of α^i dollars. Consider a specific coupon associated with product i, and assume $\mathcal{R}$ is the path (consisting of the sequence of transmissions used to distribute the coupon) over which the coupon was

transferred. Then node m gets a reward

$$\left(Q_m^i - Q_{n:(m,n)\in\mathcal{R}}^i\right)^+ \frac{\alpha^i}{\sum_{l\in\mathcal{R}}\left(Q_{s(l)}^i - Q_{r(l)}^i\right)^+}, \tag{2.10}$$

where l is a link on path $\mathcal{R}$, $s(l)$ is the sender, and $r(l)$ is the receiver. Note that this queue length information is inserted by the users before they forward the coupons to their neighbors. Furthermore, note that the amount of reward user m obtains is proportional to the queue difference. The idea is to motivate user m to send her coupon to a neighbor who has the least number of coupons and hence is most likely the one who needs the coupon. Under this scheme, the user has the motivation to follow the backpressure-like coupon distribution scheme.

A3: User Behavior:

(1) First, if node j is interested in buying product i, then user j uses all available type i coupons up to her purchasing limit b_j^i. Thus, at finer time-step t, user j purchases $y_{(j,s_i)}^i[t]$ products with coupons such that

$$\begin{aligned} y_{(j,s)}^i[t] \;=\; \min\Bigg\{ & Q_j^i[t] + \sum_{m:(m,j)\in\mathcal{L}} y_{(m,j)}^i[t], \\ & \left(b_j^i T - \sum_{\tau=0}^{t} y_{(j,s)}^i[\tau]\right)^+ \Bigg\}, \end{aligned}$$

We assume that user j never forwards type-i coupons to her neighbors if user j buys product i.

(2) If user j is not a customer buying product i, then user j needs to distribute type i coupons to her neighbors. We assume that at the beginning of finer time-step t, user j requests $Q_m^i[t]$ if user m is her neighbor, and also polls the store to obtain $\tilde{Q}_m^i[t]$. Since the amount of coupon forwarding reward is determined by

the queue difference as described in (2.10), user j selects a coupon type i^* and neighbor m^* such that

$$(i^*, m^*) \in \arg\max_{(j,m)\in\mathcal{L}} \left(Q_j^i[t] + \tilde{Q}_j^i[t] - Q_m^i[t] - \tilde{Q}_m^i[t]\right),$$

and transfers $\min\left\{\mu_j, Q_j^{i^*}[t]\right\}$ of type i^* coupons to node m^*.

Note that $\tilde{Q}_m^i[t]$ is the number of coupons used by user m but not been rewarded yet, so $\tilde{Q}_m^i[t] = 0$ if user m is not a customer buying product i. A store maintains this unrewarded coupon queue to prevent a customer receiving too many coupons. When user j uses too many coupons, the unrewarded coupon queue becomes large. After a neighbor of user j finds a large $\tilde{Q}_j^i[t]$, the neighbor realizes that user j has received too many coupons and the store might not reward him for forwarding coupons to user j. Then the neighbor will stop forwarding more coupons to user j.

A4: Coupon Generation Scheme: The coupon distributor needs to decide the number of coupons to inject into the network. We assume that coupon distributor generates μ_d type-i coupons when $Q_d^i[t] \leq Q_T q^i$, and zero type-i coupon otherwise. Here, Q_T is a constant threshold value. In other words, $\Theta^i[t] = \mu_d$ if $Q_d^i[t] \leq Q_T q^i$, and $\Theta^i[t] = 0$ otherwise.

In the following theorem, we analyze the performance of the backpressure coupon routing, and prove that

Theorem 2 *Assume that $\gamma_j^i \leq b_j^i$ for all i and j. Under the coupon management, coupon rewarding and generating scheme, and user behavior defined above, we have*

$$\lim_{Q_T\to\infty}\lim_{T\to\infty}\frac{\sum_{t=1}^{T}\Theta^i[t]}{T} = \left(\sum_{j\in\mathcal{B}_i}\check{y}_{(j,s)}^i\right), \tag{2.11}$$

and

$$\lim_{Q_T\to\infty}\lim_{T\to\infty}\frac{\sum_{t=1}^{T} y^i_{(j,s)}[t]}{T} = \check{y}^i_{(j,s)}, \tag{2.12}$$

where $\check{\mathbf{y}}$ *is the optimal solution of OPT 2.*

Proof 2 *The proof is presented in Appendix B.*

Note that although the theorem is an asymptotic result, the algorithm itself works for any value of T. A finite value of T may result in a sub-optimal solution. In our simulations, we choose $T = 300$ and the final coupon allocation is very close to the optimal.

2. Large Time Scale Control: Coupon Rate Selection

We assume that the algorithm for coupon delivery at the small time scale converges quickly to the target rate, and now consider how to choose this target rate. Recall that our means of implementing coupon delivery at rate γ^i_j is to reward the neighbors of a customer j for forwarding coupons to j at rate γ^i_j. In this section all dynamics take place at the large time. Thus, we have the sequence of target rates $\gamma^i_j[0], \cdots, \gamma^i_j[k-1], \gamma^i_j[k], \gamma^i_j[k+1], \cdots$, and the large time scale algorithm needs to guarantee that $\lim_{k\to\infty} \hat{\gamma}^i_j[k] = \hat{x}^{il}_j$.

Denote by $z^{ih}_j[k]$ and $z^{il}_j[k]$ the number of product i that user j buys from store in the interval $[k, k+1]$ at the marked price and the discounted price, respectively. Let the total number of goods purchased in the interval $[k, k+1]$ be denoted $z^i_j[k] = z^{ih}_j[k] + z^{il}_j[k]$. Further, denote the difference in purchases made by user j over intervals $[k, k+1]$ and $[k-1, k]$ by $\Delta z^i_j[k] = z^i_j[k] - z^i_j[k-1]$ corresponding to a difference in the coupon delivery rate $\Delta\gamma^i_j[k] = \gamma^i_j[k] - \gamma^i_j[k-1]$. We first intuitively understand the four possibilities associated with $\Delta\gamma^i_j[k], \Delta x^i_j[k]$ (assuming that $\Delta\gamma^i_j[k]$ is small) :

- $\Delta\gamma_j^i[k] < 0$ and $\Delta z_j^i[k] = 0$: This implies that $\gamma_j^i[k] \geq \hat{x}_j^{il}$ and the user is receiving more coupons than he can use. We need to ensure $\gamma_j^i[k+1] < \gamma_j^i[k]$.

- $\Delta\gamma_j^i[k] < 0$ and $\Delta z_j^i[k] < 0$: This implies that $\gamma_j^i[k] < \hat{x}_j^{il}$ and the user is not receiving enough coupons to realize the maximum possible number of purchases. We need to ensure $\gamma_j^i[k+1] > \gamma_j^i[k]$.

- $\Delta\gamma_j^i[k] > 0$ and $\Delta z_j^i[k] = 0$: This implies that $\gamma_j^i[k] \geq \hat{x}_j^{il}$ and the user is receiving more coupons than he can use. We need to ensure $\gamma_j^i[k+1] < \gamma_j^i[k]$.

- $\Delta\gamma_j^i[k] > 0$ and $\Delta z_j^i[k] > 0$: This implies that $\gamma_j^i[k] < \hat{x}_j^{il}$ and the user is not receiving enough coupons to realize the maximum possible number of purchases. We need to ensure $\gamma_j^i[k+1] > \gamma_j^i[k]$.

Note that an increase in the coupon rate cannot cause a decrease in the number of purchases. A simple controller that takes into account all the four possible conditions is

$$\begin{aligned} \gamma_j^i[k+1] &= (\gamma_j^i[k] + \delta)\chi_{\{\Delta\gamma_j^i[k]\Delta z_j^i[k]>0\}} \\ &\quad +(\gamma_j^i[k] - \delta)\chi_{\{\Delta\gamma_j^i[k]\Delta z_j^i[k]=0\}}. \end{aligned} \tag{2.13}$$

Here, $\delta > 0$ is a constant small amount by which we increase or decrease γ_j^i. We can now easily prove that the controller converges to within $\delta/2$ of the desired value of $\hat{\gamma}_j^i$.

Theorem 3 *Under the time scale separation assumption, using the controller (2.13) we have*

$$\lim_{k\to\infty} |\gamma_j^i[k] - \hat{x}_j^{il}| \leq \delta/2 \quad \forall\ i \in \mathcal{S},\ j \in \mathcal{R}.$$

Proof 3 *The proof is presented in Appendix C.*

Note that when δ is smaller enough and the algorithm starts with a small $\gamma_j^i[0]$, we can guarantee that $\gamma_j^i[k] \leq b_j^i$ for all k. Combining Lemma 1, Theorem 2 and Theorem 3, we conclude that *the number of coupons consumed under the two time-scale algorithm converges to the optimal solution to OPT 1.*

E. Delay-Based Coupon Forwarding

Suppose that the store rewards relays only after a coupon has been used to make a purchase. The insight that we obtain from the optimality of backpressure is the following:

- If coupons are not used on a particular path, queues build up. This would cause the average delay in being rewarded to all relays on the path to increase.
- If a higher rate of coupons than that set by the store are transferred along a path, the store does not reward the relays for some fraction of coupons and virtual coupons build up. Again, this would mean that the average delay in being rewarded to all relays on the path would increase.

The observation immediately suggests that perhaps a simpler algorithm would be to replace the backpressure based user control of Section D A3 with a much simpler scheme. Users need only keep track of the average delay experienced in obtaining rewards when they forward coupons to each of their neighbors. They choose to forward coupons to that neighbor who has the lowest such delay. The scheme is intuitively incentive compatible, since users might want to obtain rewards as soon as possible. Thus, we may replace the reward scheme of Section D A2 with an equal reward for all users in the path.

However, a few further additions are required to construct a workable heuristic algorithm. The first addition stems from the fact that under backpressure, if a user

finds that all her neighbors have larger effective queue lengths than herself, she does not forward coupons to any of them. The equivalent in the delay based regime would be to simply choose a threshold value of delay (e.g., D_U), and refuse to forward coupons to any neighbor that yields a delay larger than this threshold.

The second addition is that while keeping track of delays, even small differences in delays could result in a particular user being ignored entirely. Hence, instead of a hard comparison between the delays of different options, we soften the comparison. For example, if neighbors 1 and 2 of a node yield delays d_1 and d_2, we consider both as equally lucrative options if $|d_1 - d_2| \leq D_T$, where D_T is a constant delay threshold. Our expectation is that this simplified algorithm would perform almost as well as the backpressure-scheme.

Based on the observations above, we propose the following delay-based coupon forwarding to replace the user control of Section D A3 for all coupons in which user i is not interested.

Delay-based coupon forwarding: Consider product j that user i is not interested. Denote by $D_m^i(t)$ the average delay experienced in obtaining rewards when user j forwards type i coupons to neighbor m. User i keeps track $D_m^i(t)$ for all neighbors. At time step t, user j first selects type i^* coupon such that

$$i^* \in \arg\min_i \min_{m:(j,m)\in\mathcal{L}} D_m^i(t)$$

and a subset of neighbors associated with type i^* coupon

$$\mathcal{K}_j^{i^*} = \left\{ m : \begin{smallmatrix} \left| D_m^{i^*}(t) - \min_{m:(j,m)\in\mathcal{L}} D_m^{i^*} \right| \leq D_T \\ D_m^{i^*}(t) \leq D_U, (j,m)\in\mathcal{L} \end{smallmatrix} \right\}.$$

Then user j sends

$$\frac{\min\left\{Q_j^{i^*}(t), \mu_j(t)\right\}}{|\mathcal{K}_j^{i^*}|}$$

number of type i^* coupons to each of the neighbors in $\mathcal{K}_j^{i^*}$.

Remark: Backpressure based user control requires a user to obtain the lengths of coupon queues from her/his neighbors and from the store. Delay-based coupon forwarding, on the other hand, does not require any information exchange among the users. Each user makes decisions based on her/his own information history, which results in a much smaller communication overhead as compared to backpressure based user control. Further, unlike backpressure, the reward given to every user in the path of a coupon can be identical.

F. Simulation Results

We simulate our coupon distribution system on different network topologies to study the validity of our schemes. For the sake of comparison, we also create a third coupon distribution system in which coupons are forwarded by relays randomly to their neighbors. This would indeed be the case if multihop coupon distribution were allowed without incentives for forwarding in any particular direction. Intuitively, such a distribution scheme should over-distribute coupons, since the distributor receives no feedback. Recall that each large-scale time slot consists of T small time slots.

1. Simple Tree Topology

A simple tree topology is illustrated in Figure 5. There is a single coupon source, two relays, six leaf nodes (customers), and one store. Relays may choose one of their neighbors to forward coupons to at each time instant. Each customer j has a different value of $\hat{x}_j^l$ and $\hat{x}_j^h$. At each time instant t, users utilize all the coupons that they possess if the cumulative number of purchases made is less than $\left(\hat{x}_j^l + \hat{x}_j^h\right) T$. Once this is done, they purchase a random number of goods at the marked price, as long

as it is rational to do so (i.e., either $\sum_{\tau=0}^{t} x_j^l[t] \leq \hat{x}_j^l T$ and $\sum_{\tau=0}^{t-1} x_j^h[\tau] \leq \hat{x}_j^h T$, or $\sum_{\tau=0}^{t} x_j^l[\tau] + \sum_{\tau=0}^{t-1} x_j^h[\tau] \leq b_j^i T$ and $\sum_{\tau=0}^{t-1} x_j^h[\tau] \leq \hat{x}_j^h T$). Users repeat this process until the end of the small time period $T = 300$. At the last instant $t = T - 1$, if $\sum_{\tau=0}^{T-2} x_j^h[\tau] \leq \hat{x}_j^h T$, user j purchases $\hat{x}_j^h T - \sum_{\tau=0}^{T-2} x_j^h[\tau]$ goods.

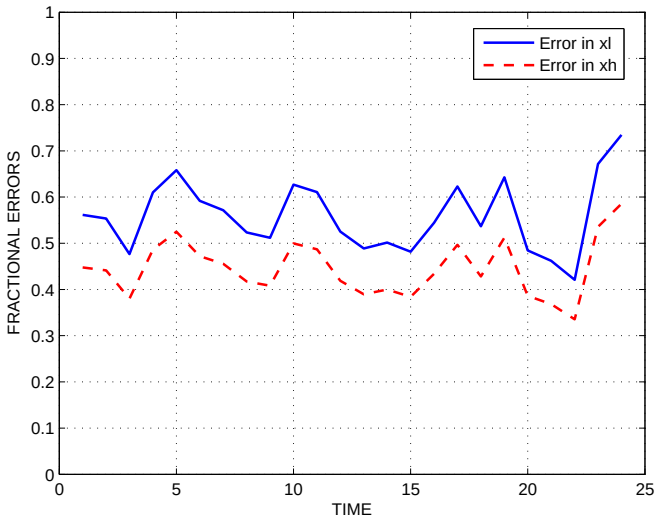

Fig. 2. Example trajectories of fractional errors, random distribution

We first verify that the small time scale dynamics of backpressure is able to distribute the correct number of coupons to any user j. The capacities of all the relay links are set to 300 coupons per unit time. We illustrate the trajectory of purchases made by user 3 who has $\hat{x}_3^l = 50$ and $\hat{x}_3^h = 60$ over a time interval $T = 300$ units in Figure 6. For purposes of illustration, we assume that $\gamma_3 = \hat{x}_3^l = 50$ In other words, we set the reward rate for coupon forwarding by neighbors of user 3 exactly equal to the average rate at which the user 3 should be given coupons in order to extract maximum revenue. We see in Figure 6 that the backpressure scheme indeed gives the right number (and rate) of coupons to the user, ensuring that $x_3^l[T] = 50$ and

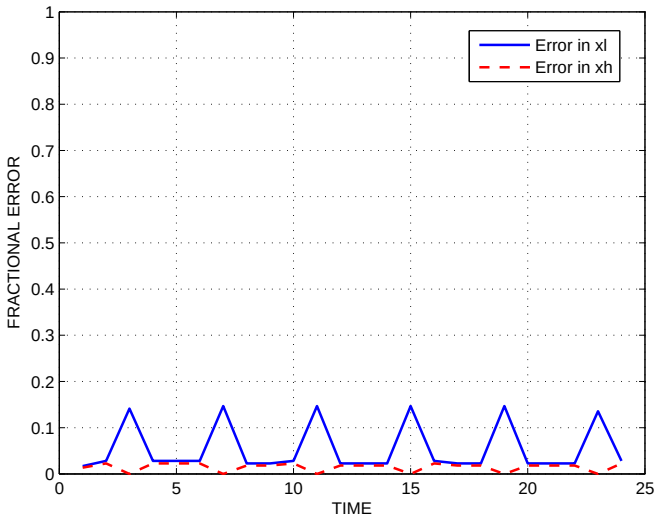

Fig. 3. Example trajectories of fractional errors, backpressure distribution

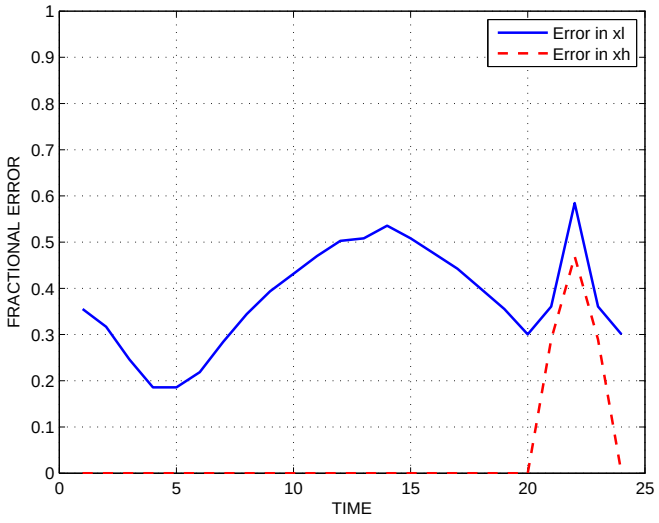

Fig. 4. Example trajectories of fractional errors, delay distribution

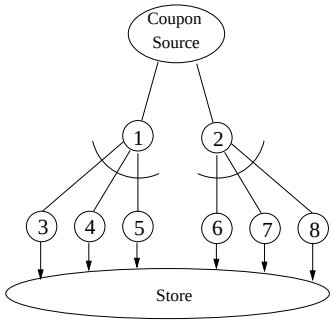

Fig. 5. Simple tree topology.

$x_3^h[T] = 60$.

We now simulate all three schemes (small and large time scales) and the results are as follows. Figure **??** shows the fractional error in high and low price purchases made by user 3, as compared to $\hat{x}_3^h$ and $\hat{x}_3^l$ for the delay based-scheme. We notice that there is a significant error. We plot the same quantities when we use the backpressure-scheme in Figure **??**. The scheme quickly converges, causing the errors to be small. Finally, we plot the same for the delay-based scheme in Figure **??**. For this scheme, we chose the cut-off threshold to be $T/8$ and the acceptable delay difference to be 15%. Its error is between the other schemes.

Finally, we plot the total revenue obtained by the store for the two different schemes, and compare them to the maximum possible revenue in Figure 7. The upper bound is the value $\sum_j p\hat{x}_j^h + q\hat{x}_j^l$, which is the maximum extractable revenue. We see that the randomized algorithm does significantly worse than backpressure as well as delay-based schemes. Even accounting for the fact that a constant part of the revenue would have to be used to incentivize the scheme, the performance improvement is still significant, although the delay-based scheme performs worse than backpressure.

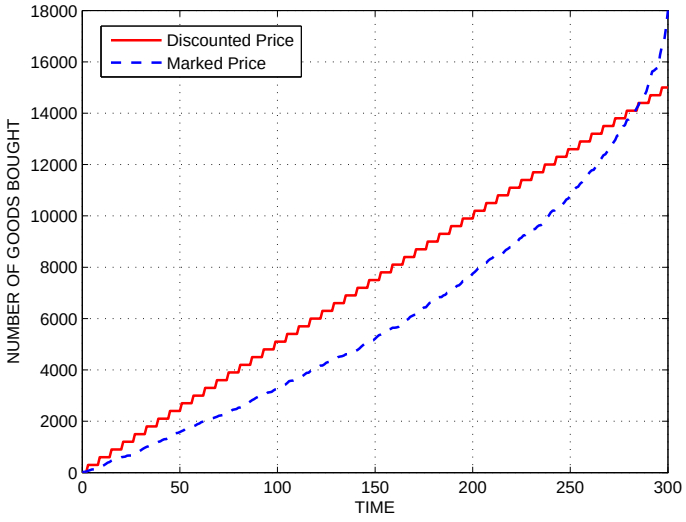

Fig. 6. An example trajectory for user 3 over the small time scale. The solid line indicates purchases made at the marked price, while the dashed line indicates discounted purchases. The user has $\hat{x}_j^l = 50$ and $\hat{x}_j^h = 60$. In this example we have set (for illustration) $\gamma_j = \hat{x}_j^l = 50$, i.e., the reward rate for coupon forwarding is known exactly, and we see that the user receives exactly the right number of coupons.

2. Power Law Topology

We now consider a power-law topology that might bear a closer resemblance to real-world social networks. The graph consists of 100 nodes, and is constructed using preferential-attachment [14] with each entering node connecting to two others. Nodes are connected to the coupon source, are relays, and are customers with probabilities 0.2, 0.7 and 0.1, respectively. Customers have arbitrary spending capacities. We show the upper bound and the performance of the three schemes in Figure 8. Back-pressure clearly performs the best, followed by the delay-based and random schemes. The results indicate that using such coupon distribution schemes could significantly

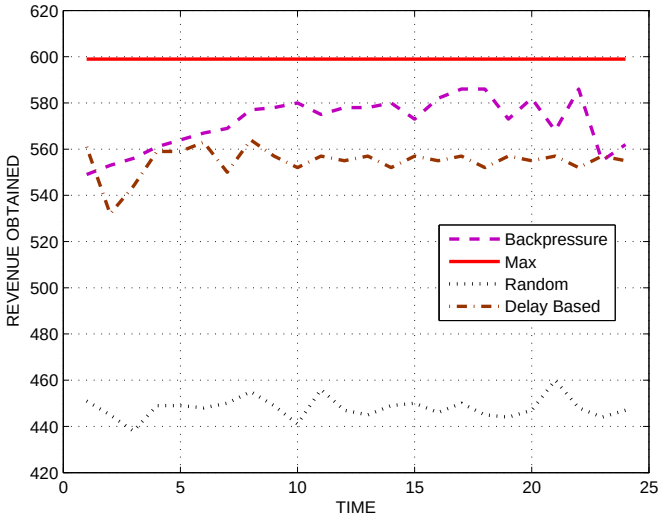

Fig. 7. Trajectory of revenue obtained by the store using different schemes.

increase the revenue obtained.

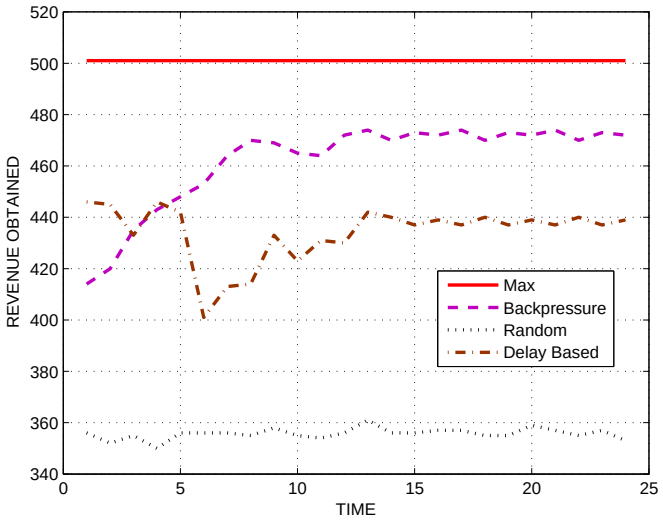

Fig. 8. Trajectory of revenue obtained by the store using different schemes for the power-law topology.

CHAPTER III

VALUE-AWARE RESOURCE ALLOCATION FOR SERVICE GUARANTEES IN NETWORKS

A. Main Results

Classical optimization-decomposition techniques usually yield a "source-rate responds to link-price" type of controller [15, 17–19], wherein each link's price increases with the link-load in order to prevent the link-capacity from being exceeded. As the link-price increases, sources cut down their transmission rates, where the aggressiveness of the source controller is determined by its utility function. However, the solution to our problem has remained elusive due to strong coupling between the quality seen at source that requires a hard guarantee, and the *link quality degradation* that depends on the link-loads along its route.

We first present some examples of what the quality degradation functions might look like in Section B, and discuss examples that we use later in this work. We then proceed to provide a centralized solution to the problem in Section C. We develop two algorithms to this end. A primal algorithm is presented in Section D. Our main contribution, a dual algorithm is presented in Section E, and stems from the realization that it is possible to decouple the QoS guarantees at sources and link quality degradation using *effective capacity* that is based on the link-price and user-dissatisfaction. Once we choose an effective capacity for a link, the quality degradation depends solely on this choice, and not on the actual link-load.

Each source declares its *dissatisfaction* to the links it uses based on the difference between the quality it sees and what it requires. The links set a *price* based on the difference between load and *effective capacity*, which in turn depends on link-load

and total user dissatisfaction. This decoupling of link-load and effective capacity appears to have the correct properties to allow distributed solution. Finally, sources use route-price (the sum of all link-prices on a route) to determine the source-rate.

We prove that the algorithm is indeed capable of solving our resource allocation problem using Lyapunov techniques [31]. We illustrate the optimality of the solution reached by our distributed dual control scheme in some selected cases by directly solving the optimization problem. We simulate the controller and conduct experiments on realistic topologies in Section F. We conclude with pointers to future work in Section **??**.

B. Examples of Quality Degradation Functions

We first begin with some ideas of link quality degradation with load. We make several assumptions on the properties of link quality degradation functions. Mathematically, one can list them in the following manner. If the total sum-rate on any link is $y_l = \sum_{r \in \mathcal{S}} R_{lr} x_r$ then

- the quality degradation function $V_l(y_l)$ is non-negative and convex increasing in link-load y_l,
- the total quality degradation seen by flow r is $\sum_{l \in \mathcal{L}} R_{lr} V_l(y_l)$ (i.e., quality degradation sums up over multiple hops), and
- finally, in our deterministic approach to the problem, we have an implicit assumption that the service process at one link does not impact the arrival process at the succeeding link.

While the above assumptions result in mathematical simplicity of our optimization problem, we believe that they provide acceptable models of quality degradations

in communication systems with queues. Below, we justify our assumptions with some common examples of quality degradation functions.

1. Average Delay in M/M/1 Queues

For an M/M/1 queue with arrival rate x and service rate c, the expected waiting time in the queue is $\frac{x}{c(c-x)}$ for a stable queue, that is when $x < c$. In this case, one can write quality degradation function to be the expected waiting time for any packet in the queue. That is,

$$V(x) = \frac{x}{c\,(c-x)}.$$

We note that the quality degradation function is always positive and increases from 0 to ∞ when x ranges in $[0, c)$. Further,

$$V'(x) = \frac{1}{(c-x)^2} > 0,$$
$$V''(x) = \frac{2}{(c-x)^3} > 0.$$

Hence, the function is convex increasing.

2. Decay-rate of a Fluid Buffer With On-Off Service Process

Consider a single server queue with constant-rate arrival x and a two-state On-Off service process where on and off times are exponentially distributed with rates μ and λ respectively. When service is on and the buffer is non-empty, it is serviced at a constant rate $R > x$ such that $x < R\frac{\lambda}{\lambda+\mu}$. It can be shown [32] that the probability of buffer exceeding a threshold z is exponentially decreasing and a possible quality degradation function in this case could be the inverse of this decay-rate. One can write down this decay-rate explicitly in terms of the above parameters as

$$V(x) = \left(-\lim_{z\to\infty} \frac{\log Pr\left\{L > z\right\}}{z} \right)^{-1} = \left(\frac{\lambda}{x} - \frac{\mu}{R-x} \right)^{-1}.$$

If we denote $R\frac{\lambda}{\lambda+\mu}$ by c then, one can write

$$V(x) = \frac{x\left(\frac{c}{\lambda} - \frac{x}{\mu+\lambda}\right)}{c-x}.$$

First, we notice that $V(x)$ is always non-negative in the interval $[0, c)$. Second, we see that when x approaches 0, the value $V(x)$ diminishes to zero. Analogously, when x approaches c, the value $V(x)$ tends to infinity. Additionally,

$$V'(x) = \frac{1}{\mu+\lambda}\left(1 + \frac{\mu}{\lambda}\left(\frac{c}{c-x}\right)^2\right) > 0,$$
$$V''(x) = \frac{2c^2\mu}{\lambda(\mu+\lambda)(c-x)^3} > 0.$$

Hence, one can conclude it is convex increasing. Recent results [33] suggest that under appropriate conditions, even when packets of a flow traverse from one queue to the next in a network, the delay seen in each queue is independent of the others.

C. Centralized Resource Allocation

We start by developing ideas on how to solve our resource allocation problem in a centralized fashion and creating model networks that we will use as examples to illustrate the performance of different control loops throughout this work. We recall our optimization problem, repeated here for convenience,

$$\max \sum_{r \in \mathcal{S}} U_r(x_r) \tag{3.1}$$

subject to the constraints

$$\begin{aligned} y_l &\leq c_l, \ \forall \ l \in \mathcal{L} \\ \sum_{l \in \mathcal{L}} R_{lr} V_l \left(y_l \right) &\leq \sigma_r, \ \forall \ r \in \mathcal{S} \\ x_r &\geq 0, \ \forall \ r \in \mathcal{S}. \end{aligned} \tag{3.2}$$

Consider the case, where utility functions assume unbounded negative values when $x_r = 0$ and quality degradation functions grow unbounded when sum-rate y_l approach c_l. Then, clearly $x_r > 0$ and $y_l < c_l$ for optimal solution. Let x_r^* be a feasible point and there exist constants $w_r \geq 0$ such that

$$\begin{aligned} U_r'(x_r^*) - \sum_{s \in \mathcal{S}} w_s \sum_{l \in \mathcal{L}} R_{ls} R_{lr} V_l' \left(\sum_{i \in \mathcal{S}} R_{li} x_i^* \right) = 0, \ \forall \ r \in \mathcal{S} \\ w_r \left(\sum_{l \in \mathcal{L}} R_{lr} V_l \left(\sum_{i \in \mathcal{S}} R_{li} x_i^* \right) - \sigma_r \right) = 0, \ \forall \ r \in \mathcal{S}, \end{aligned} \tag{3.3}$$

then x_r^* is a global maximum and if U_r is strictly concave, then x_r^* is unique global maximum.

We will illustrate by the following examples how our model takes into consideration all of the desired properties of the quality degradation function and how they impact resource allocation with service guarantees.

1. Homogeneous Tandem Network

Consider n flows sharing a simple tandem network of L links where each link has a constant capacity c and identical quality degradation function V_l associated with each link. All the flows originate at the first node and their destination is the final node as shown in Fig. 9.

We assume that associated with each flow i is a utility function $a_i \log(x_i)$ corresponding to its throughput x_i, and a service guarantee σ_i. We also take the quality

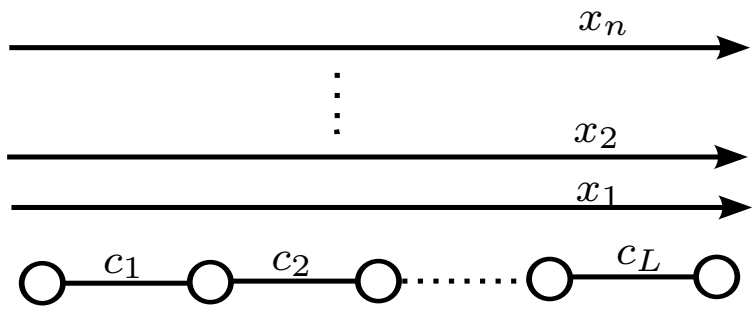

Fig. 9. Tandem network of L links being shared by n flows.

degradation function to be $V_l(y_l) = -\log(1 - \frac{y_l}{c_l})$. We choose this quality degradation function, as it satisfies the desired convexity property. It also captures the effect that for achieving capacity over erroneous communication links, one needs to use arbitrary long codes leading to unbounded variance in available service. Thirdly and very importantly, this choice of quality degradation function gives us analytical expression for x_i's, that offers valuable insight into trade-off between throughput and service guarantees. Under these assumptions, we have the following optimization problem

$$\max \quad \sum_{i=1}^{n} a_i \log x_i \tag{3.4}$$

subject to the constraints

$$\begin{aligned} -L\log(1 - \frac{\sum_{i=1}^{n} x_i}{c}) &\leq \sigma_i, \quad i = 1, 2, \ldots, n \\ \sum_{i=1}^{n} x_i &\leq c_l = c, \quad l = 1, 2, \ldots, L \\ x_i &\geq 0, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{3.5}$$

The log function ensures that all x_i's are always positive and also that the sum-rate constraints are never active. Let w_i be the Lagrange multipliers associated with

the quality degradation constraint, then the Lagrangian is given by

$$L(x,w) = \sum_{i=1}^{n} a_i \log x_i + \sum_{j=1}^{n} w_j \left(L \log \left(1 - \frac{\sum_{k=1}^{n} x_k}{c} \right) + \sigma_j \right) \tag{3.6}$$

Differentiating the Lagrangian with respect to x_i's and equating them to zero, we obtain

$$c - \sum_{j=1}^{n} x_j^* = \frac{L \sum_{k=1}^{n} w_k}{a_i} x_i^*. \tag{3.7}$$

Defining row vector $\mathbf{1}$ of ones of size n, and $n \times n$ matrix D with real entries such that $D_{jj} = 1 + \frac{L \sum_{k=1}^{n} w_k}{a_j}$ and $D_{ij} = 1$ for $i \neq j$; we can equivalently write the above equation in the following compact form,

$$x^* \mathbf{D} = c\mathbf{1}. \tag{3.8}$$

Let $i^* = \arg\min\{\sigma_i : i = 1, 2, \ldots, n\}$, then $w_i = 0, i \neq i^*$ and $c - \sum_{j=1}^{n} x_j^* = c \exp(-\sigma_{i^*}/L)$ by KKT conditions. Then,

$$x_i^* = \left(a_i / \sum_{i=1}^{n} a_i \right) c \left(1 - \exp(-\sigma_{i^*}/L) \right).$$

This example verifies that our modeling intuition is right for resource allocation with service guarantees. Here are some observations from this simple example:

- For any finite service requirement, sum-rate is always less than the capacity of each link.
- Throughputs decrease with number of hops due to service requirements.
- When quality degradation is inherent to a link, flow with most stringent service requirements limits the throughput for every other flow.

2. Three-Flows Two-Hop Network

Consider the network in Fig. 10 in which three sources transmit over two links. Let link i have capacity c_i. Assuming log utility and quality degradation functions as in previous example, the resource allocation problem becomes

$$\max \quad \sum_{i=1}^{n} a_i \log x_i \tag{3.9}$$

subject to the constraints

$$\begin{aligned} -\log(1 - \frac{x_i + x_3}{c_i}) &\leq \sigma_i, \quad i = 1, 2 \\ -\sum_{i=1}^{2} \log(1 - \frac{x_i + x_3}{c_i}) &\leq \sigma_3. \end{aligned} \tag{3.10}$$

Let w_i be the Lagrange multipliers corresponding to quality degradation constraint

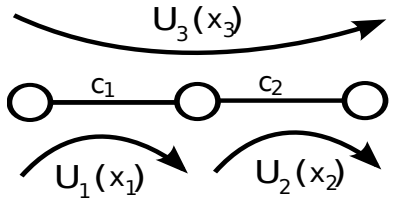

Fig. 10. Three flows sharing a two-link network.

of flow i, then the *Lagrangian* is written

$$\begin{aligned} L(x, w) = &\sum_{i=1}^{3} a_i \log x_i + \sum_{i=1}^{2} w_i \left(\log(1 - \frac{x_i + x_3}{c_i}) + \sigma_i \right) \\ &+ w_3 \left(\sum_{i=1}^{2} \log(1 - \frac{x_i + x_3}{c_i}) + \sigma_3 \right) \end{aligned} \tag{3.11}$$

From above equations, we can derive the KKT conditions

$$\begin{aligned}
\frac{a_i}{x_i^*} - \frac{w_i + w_3}{c_i - x_i^* - x_3^*} &= 0, \quad i = 1, 2 \\
\frac{a_3}{x_3^*} - \sum_{i=1}^{2} \frac{w_i + w_3}{c_i - x_i^* - x_3^*} &= 0, \\
w_i \left(\log(1 - \frac{x_i + x_3}{c_i}) + \sigma_i \right) &= 0, \quad i = 1, 2 \\
w_3 \left(\sum_{i=1}^{2} \log(1 - \frac{x_i + x_3}{c_i}) + \sigma_3 \right) &= 0.
\end{aligned} \tag{3.12}$$

Let us consider the case when $\sigma_3 < \min\{\sigma_1, \sigma_2\}$. In this case, $w_1 = w_2 = 0$ and $\left(\sum_{i=1}^{2} \log(1 - \frac{x_i + x_3}{c_i}) + \sigma_3 \right) = 0$. For the simple case of $a_i = 1,\ i = 1, 2, 3$ we have optimal rates

$$x_1^* = x_2^* = 2x_3^* = \frac{2c}{3} \left(1 - \exp(-\sigma_3/2)\right). \tag{3.13}$$

This example verifies our modeling intuition for resource allocation with service guarantees, with same conclusions as in previous example. We will use above two simple examples for numerical studies of our dual algorithm in Section E.

D. Primal Algorithm

We now develop an algorithm that could potentially be used to obtain an approximate solution of our optimization problem. The approach that we use is called the *Primal* method, as it follows from the Primal formulation of the problem. The main idea is to relax the constraints by incorporating them as a cost into the objective. Essentially, the idea is that there is a price to violating the quality constraints, and we maximize utility minus price. Thus we consider the function

$$J(x) = \sum_{r \in \mathcal{S}} \left(U_r(x_r) - B_r \left(\sum_{l \in \mathcal{L}} R_{lr} V_l \left(y_l \right) \right) \right), \tag{3.14}$$

where $B_r(\cdot)$ is a barrier function assumed to be convex increasing, from zero to unbounded values when argument increases from zero to σ_r. To minimize this function, we can use a gradient descent approach, i.e.,

$$\begin{aligned} \dot{x}_r &= k_r(x_r)\left(U_r'(x_r) - q_r\right), \\ q_r &= \sum_{s\in\mathcal{S}} B_s'\left(\sum_{l\in\mathcal{L}} R_{ls}V_l\left(y_l\right)\right)\sum_{l\in\mathcal{L}} R_{ls}R_{lr}V_l'\left(y_l\right). \end{aligned} \tag{3.15}$$

Since the problem is convex, it is straightforward to show using Lyapunov techniques [16, 23, 31] that the above algorithm converges, and leads to one maximizer of (3.14). To this end, note that $J(x)$ as defined in (3.14) is a strictly concave function. We denote its unique maximizer by $\hat{x}$. Then, $J(\hat{x}) - J(x)$ is non-negative and equals zero only at $x = \hat{x}$. This makes $W(x) \triangleq J(\hat{x}) - J(x)$, a natural candidate Lyapunov function and we use it in the following proposition, which has a similar proof to that of [23].

Proposition 4 *Consider a network in which all sources follow the primal control algorithm* (3.15). *Let* $J(x)$ *be as defined in* (3.14) *and functions* $U_r(\cdot), k_r(\cdot), V_l(\cdot)$ *and* $B_s(\cdot)$ *be such that* $W(x)$ *grows unbounded as* $\|x\| \to \infty$, *and* $\hat{x}_i > 0$ *for all* i. *Then, the controller in* (3.15) *is globally asymptotically stable and the equilibrium value maximizes* (3.14).

Proof 4 *Differentiating* $W(x)$ *with time* t, *we get*

$$\begin{aligned} \dot{W} &= -\sum_{r\in\mathcal{S}} \frac{\partial J}{\partial x_r}\dot{x}_r \\ &= -\sum_{r\in\mathcal{S}} k_r(x_r)\left(U_r'(x_r) - q_r\right)^2 < 0, \quad \forall x \neq \hat{x}, \end{aligned}$$

and $\dot{W} = 0$ *for* $x = \hat{x}$. *Here, second line of the equation follows from equations* (3.14) *and* (3.15). *Thus, all the conditions of the Lyapunov theorem are satisfied and we*

have proved that the system state converges to $\hat{x}$.

However, primal controller suffers from the following handicaps. The approach is not optimal since the relaxation would yield an acceptable solution only if the barrier values at the optimal solution of our original objective (3.1) were small. Further, the above formulation would not allow for optimal points on the boundary of constraint set. We now approach the problem from a dual perspective to see if we can obtain any better insight.

E. Dual Algorithm

We start by writing down *Dual* version of our resource allocation problem defined in (3.1) in the hope it yields insight on how to obtain a distributed method of achieving optimal resource allocation. The Dual problem corresponding to the problem defined in (3.1) is given by

$$D(w) = \max_{x_s \geq 0} \sum_{s \in \mathcal{S}} U_s(x_s) - w_s \left(\sum_{l \in \mathcal{L}} R_{ls} V_l \left(y_l \right) - \sigma_s \right).$$

Let x_r^* be the optimal maximizer, then

$$U_r'(x_r^*) = \sum_{s \in \mathcal{S}} w_s \sum_{l \in \mathcal{L}} R_{ls} R_{lr} V_l' \left(y_l^* \right).$$

This gives us a system of equations that needs to be solved to find the optimal x_r^* for each w. However, this is in an implicit form that requires the knowledge of load on every link that a flow traverses. Therefore, this approach is not completely distributed.

Nevertheless, this formulation gives us the hint that instead of link load and link-degradation being dependent on each other directly with load $y = Rx$ and degradation $V(y)$, we could break up their coupling. We do this by introducing a new variable $\tilde{y}$

that we refer to as *effective capacity*. The relaxed version of the resource allocation problem is now

$$\max \sum_{r \in \mathcal{S}} U_r(x_r) \tag{3.16}$$

subject to the constraints

$$\begin{aligned} \sum_{r \in \mathcal{S}} R_{lr} x_r = y_l \leq \tilde{y}_l, \ \forall \ l \in \mathcal{L} \\ \sum_{l \in \mathcal{L}} R_{lr} V_l \left(\tilde{y}_l \right) \leq \sigma_r, \ \forall \ r \in \mathcal{S} \\ x_r \geq 0, \ \forall \ r \in \mathcal{S}. \end{aligned} \tag{3.17}$$

Assuming that our concave utility and convex quality degradation functions ensure that respectively x_r's and $(c_l - \tilde{y}_l)$'s are always positive, we can express the Dual problem in terms of positive Lagrange multipliers p_l's and w_r's

$$\min_{p,w \geq 0} D(p, w) \tag{3.18}$$

where $D(p, w)$ is the maximum of the *Lagrangian* $L(x, \tilde{y}, p, w)$ with respect to $x, \tilde{y}$

$$\begin{aligned} D(p, w) = \max_{x_s, \tilde{y}_l \geq 0} \sum_{s \in \mathcal{S}} U_s(x_s) - \sum_{l \in \mathcal{L}} p_l \left(\sum_{s \in \mathcal{S}} R_{ls} x_s - \tilde{y}_l \right) \\ - \sum_{s \in \mathcal{S}} w_s \left(\sum_{l \in \mathcal{L}} R_{ls} V_l \left(\tilde{y}_l \right) - \sigma_s \right). \end{aligned}$$

Let $x^*, \tilde{y}^*$ be the maximizers for the above problem for any p, w, then

$$\begin{aligned} U_r'(x_r^*) = \sum_{l \in \mathcal{L}} R_{lr} p_l, \\ p_l = V_l'(\tilde{y}_l^*) \sum_{r \in \mathcal{S}} R_{lr} w_r. \end{aligned} \tag{3.19}$$

Now, we find the partial derivatives of $D(p, w)$ with respect to variables p and w

$$\begin{aligned} \frac{\partial D}{\partial p_l} &= \tilde{y}_l^* - \sum_{s \in \mathcal{S}} R_{ls} x_s^*, \quad l \in \mathcal{L} \\ \frac{\partial D}{\partial w_r} &= \sigma_r - \sum_{l \in \mathcal{L}} R_{lr} V_l\left(\tilde{y}_l^*\right) \quad r \in \mathcal{S}. \end{aligned} \tag{3.20}$$

Then the update equations for solving the Dual minimization of the relaxed problem are

$$\begin{aligned} \dot{p}_l &= h_l(p_l) \left(\sum_{s \in \mathcal{S}} R_{ls} x_s^* - \tilde{y}_l^* \right)_{p_l}^+, \quad l \in \mathcal{L} \\ \dot{w}_r &= k_r(w_r) \left(\sum_{l \in \mathcal{L}} R_{lr} V_l\left(\tilde{y}_l^*\right) - \sigma_r \right)_{w_r}^+ \quad r \in \mathcal{S}, \end{aligned} \tag{3.21}$$

where $h_l(\cdot), k_r(\cdot)$are positive functions and the notation $(z)_\rho^+$ is used to denote the function

$$(z)_\rho^+ = \begin{cases} z & \rho \geq 0 \\ \max\{z, 0\} & \rho = 0. \end{cases}$$

1. Distributed Algorithm

We can now easily see that the above algorithm is distributed in nature. At any time during evolution of our algorithm, we can treat Lagrange multipliers p_l and w_r as link-price and route-dissatisfaction, respectively. A flow r needs to "pay" link-price p_l for congesting link l if it uses the link (with the route-price being the sum of all such p_ls), and w_r is its end-to-end dissatisfaction under the current system state. The effective capacity of link l is $\tilde{y}_l$ and is decoupled from the actual load on this link $y_l = \sum_{r \in \mathcal{S}} R_{lr} x_r$. We denote the sum of link-prices by $q_r = \sum_{l \in \mathcal{L}} R_{lr} p_l$ for any flow r, and sum of route-dissatisfaction on a link l by $\nu_l = \sum_{r \in \mathcal{S}} R_{lr} w_r$ to yield the total dissatisfaction on that link. Note that such a total implies that there is no need to maintain per-flow information at the link. We denote the effective quality

degradation seen by any flow r by $\tilde{\sigma}_r = \sum_{l \in \mathcal{L}} R_{lr} V_l(\tilde{y}_l)$. Notice again that due to decoupling through $\tilde{y}$, the perceived quality degradation is a function of effective capacity, and not the actual link-load.

The algorithm is illustrated in Fig. 11. Although the diagram is reminiscent of traditional "source-rate responds to link-price" [15–18] corresponding to the congestion control problem defined in (1.1), the system is actually very different. The system may be described as follows:

- Each flow r, as it traverses its route, it accrues the price q_r that it needs to "pay" for using each of the links l. Using this route-price, each source computes a feasible rate

  $$x_r = U_r'^{-1}(q_r).$$

  Furthermore, each source declares its *dissatisfaction* w_r to the links it uses based on the difference between the quality degradation $\tilde{\sigma}_r$ that it sees and σ_r what it is willing to tolerate. The dissatisfaction is updated using

  $$\dot{w}_r = k_r(w_r) \left(\tilde{\sigma}_r - \sigma_r\right)_{w_r}^{+}.$$

- Each link detects the total dissatisfaction ν_l of flows sharing it and computes effective capacity

  $$\tilde{y}_l = V_l'^{-1}(p_l/\nu_l)$$

  and updates the link price by

  $$\dot{p}_l = h_l(p_l) \left(y_l - \tilde{y}_l\right)_{p_l}^{+}.$$

  Further, the link ensures that the quality degradation suffered by sharing flows is $V_l(\tilde{y}_l)$ by adding to or dropping part of the total flow as needed.

In summary, along with the two traditional elements of source-rate x_r and link-price

p_l, we have two additional control variables: source-dissatisfaction w_r and effective capacity $\tilde{y}_l$ (with link-degradation $V_l(\tilde{y}_l)$) that provide two further dimensions of control that are required for distributed solution.

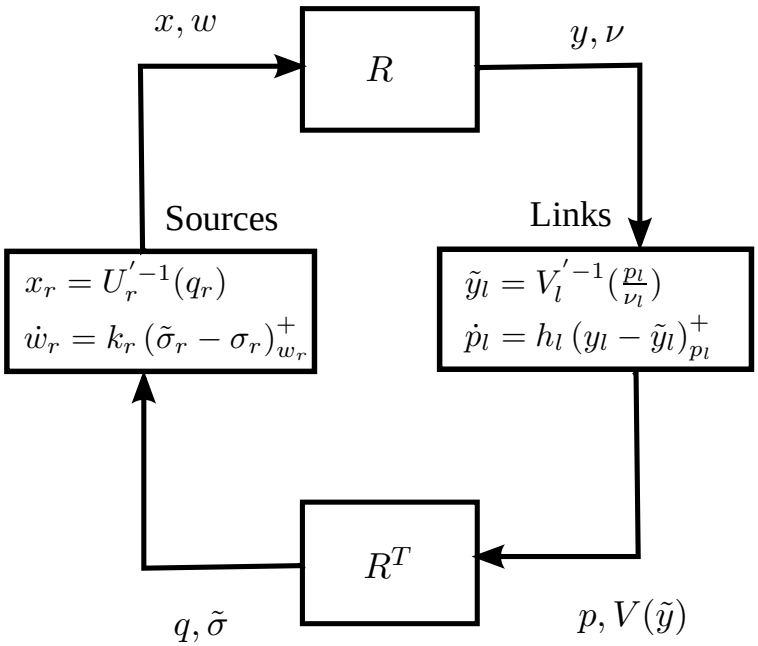

Fig. 11. A block diagram of value-aware resource allocation that allows decoupling of user-dissatisfaction on the source side, and quality on the link side.

Now, we show that under reasonable assumptions over $V_l(\cdot)$, the slackness condition of sum-rate being less than or equal to *effective capacity* is always satisfied with equality at equilibrium.

Proposition 5 *Let us assume that $V_l(\cdot)$ is strictly convex and increasing. Then, at the equilibrium $y_l = \tilde{y}_l$ for all $l \in \mathcal{L}$.*

Proof 5 *Our proof is by contradiction. Let us assume that there is a $l \in \mathcal{L}$, such that $y_l < \tilde{y}_l$. Then for this l, we have $p_l = 0$. Note that $V_l'(\cdot)$ is a non-negative increasing function. That is, either $V_l'(0) = 0$ or $V_l'(z) > 0$ for all $z \in [0, c_l]$. For the former*

case, $0 \leq y_l \leq \tilde{y}_l = 0 = V_l^{'-1}(0)$*, i.e.,* $y_l = \tilde{y}_l$*. For the latter case,* p_l *cannot be zero since* $V_l^{'-1}(0)$ *is not in the feasible range of* y_l *and hence this will force* $y_l = \tilde{y}_l$ *at the equilibrium.*

We have in effect, shown that the equilibrium conditions of our control loop satisfy the KKT conditions of our original optimization problem defined in (3.1). The conditions are easy to verify, and we may state this result as a corollary of Proposition 5.

Corollary 6 *The stationary point of* (3.21) *is a maximizer of the convex optimization problem described by* (3.1)*.*

2. Global Stability of Distributed Algorithm

It is quite easy to show that the above algorithm is globally asymptotically stable. To show this, we choose our Lyapunov function to be

$$Q(p, w) = D(p, w) - D(\hat{p}, \hat{w}), \tag{3.22}$$

where $\hat{p}, \hat{w}$ are the unique minimizers of $D(p, w)$. It is clear that $Q(p, w) \geq 0$ for all values of p, w. Also, it is easily seen that $D(p, w)$ grows radially unbounded in p, w for our choice of $V_l(\cdot)$. Therefore, to show that the above algorithm is stable it suffices to show $\dot{D}(p, w) \leq 0$, with equality iff $p = \hat{p}$ and $w = \hat{w}$. Note that at $\hat{p}, \hat{w}$, one would have $\dot{p}_l = \dot{w}_r = 0$.

Proposition 7 *Let* $Q(p, w)$ *be as defined in* (3.22) *and functions* $U_r(\cdot), V_l(\cdot), k_r(\cdot)$ *and* $h_l(\cdot)$ *be such that* $Q(p, w)$ *grows unbounded with* $\|p\|$ *and* $\|w\|$*. Then the controller in* (3.21) *is globally asymptotically stable and the equilibrium value maximizes* (3.1)*.*

Proof 6 *Differentiating D with respect to time, we get*

$$\begin{aligned}\dot{D}(p,w) &= \sum_{l\in\mathcal{L}} \frac{\partial D}{\partial p_l}\dot{p}_l + \sum_{r\in\mathcal{S}} \frac{\partial D}{\partial w_r}\dot{w}_r \\ &= -\sum_{l\in\mathcal{L}} h_l(p_l)\left(y_l - \tilde{y}_l\right)\left(y_l - \tilde{y}_l\right)^+_{p_l} \\ &- \sum_{r\in\mathcal{S}} k_r(w_r)\left(\tilde{\sigma}_r - \sigma_r\right)\left(\tilde{\sigma}_r - \sigma_r\right)^+_{w_r} \\ &< 0, \quad \forall p, w \neq \hat{p}, \hat{w},\end{aligned}$$

and $\dot{D}(\hat{p},\hat{w}) = 0$. Here, the second line of equation follows from equations (3.20) *and* (3.21)*. Thus, all the conditions of the Lyapunov theorem [31] are satisfied and we have proved that the Lagrange multipliers converge to $\hat{p},\hat{w}$. Hence, the system converges to the minimizer of* (3.18)*. From the convexity of our original problem* (3.1)*, and Corollary 6, this in turn implies that the stable point is the maximizer of* (3.1)*.*

Now, we study our primary examples of homogeneous tandem network and three-flows two-links network to compare our dual algorithm's performance with the optimal solution.

3. Homogeneous Tandem Network

Consider the same setting as in 1. Then, for an optimal x^*, y^* we would have

$$\begin{aligned}x_i^* &= \frac{a_i}{Lp} \\ y^* &= c - \frac{\sum_{i=1}^n w_i}{p}.\end{aligned}$$

Here, we have assumed that $p_l = p$ and $y_l^* = y^*$ by symmetry. We would also have additional equations from KKT conditions

$$p\left(\sum_{i=1}^{n} x_i^* - y^*\right) = 0$$
$$w_i\left(L\log\left(1 - \frac{y^*}{c}\right) + \sigma_i\right) = 0.$$

Finiteness of x_i^* from throughput and quality degradation constraint ensures that $p \neq 0$ and the solution degenerates to the solution of original resource allocation. We can also verify that by simulating dual algorithm on Simulink and using our proposed distributed algorithm to find the optimal operating point as predicted. This exercise also shows that proposed algorithm indeed converges.

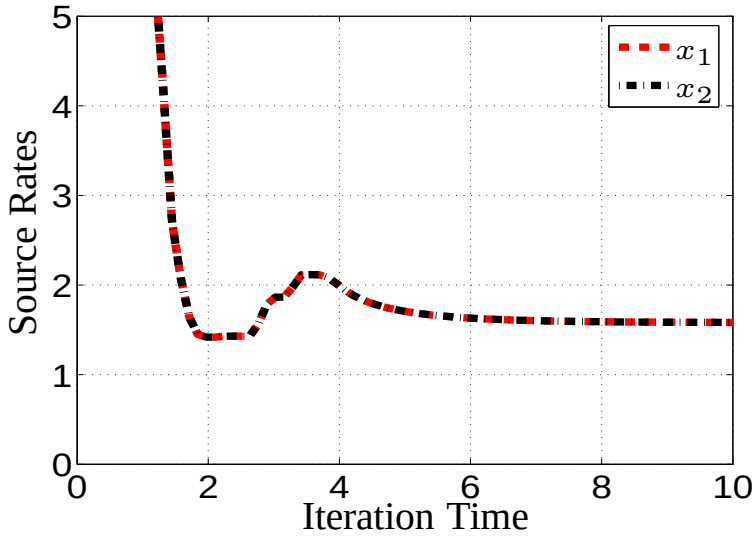

Fig. 12. Distributed resource allocation for homogeneous tandem network.

We used the following parameters for our Simulink model, $n = 2, L = 3, \sigma_1 = 24, \sigma_2 = 3, c_1 = c_2 = c_3 = 5$ and $a_1 = a_2 = 1$. For this case, we have $x_1^* = x_2^* = 2.5 * (1 - \exp(-1)) = 1.5803$. We have plotted the convergence of source rates with

iteration time for both the flows in Fig. 12.

4. Three-Flows Two-Hop Network

Consider the same setting as in 2. Then, for an optimal x^*, y^* we would have

$$x_i^* = \frac{1}{p_i}, \quad i = 1, 2 \qquad x_3^* = \frac{1}{p_1 + p_2}$$
$$y_i^* = c - \frac{w_i + w_3}{p_i}, \quad i = 1, 2.$$

Here, we have assumed that $p_l = p$ and $y_l^* = y^*$ by symmetry. We would also have additional equations from KKT conditions

$$p\left(\sum_{i=1}^{n} x_i^* - y^*\right) = 0$$
$$w_i\left(\log\left(1 - \frac{y_i^*}{c}\right) + \sigma_i\right) = 0, \quad i = 1, 2$$
$$w_3\left(\sum_{i=1}^{2} \log\left(1 - \frac{y_i^*}{c}\right) + \sigma_3\right) = 0.$$

Finiteness of x_i^* from throughput and quality degradation constraint ensures that $p \neq 0$ and the solution degenerates to the solution of original resource allocation problem since $y_i^* = x_i^* + x_3^*, \quad i = 1, 2$.

For numerical study of convergence of our proposed algorithm for this topology, we used the following parameters in our Simulink model, $\sigma_1 = \sigma_2 = 1, \sigma_3 = 3, c_1 = c_2 = 5$ and $a_i = 1, \quad i = 1, 2, 3$. For this case, $\sigma_3 > \sigma_1 + \sigma_2$, and hence we would have $w_3 = 0$. Therefore, $x_1^* = x_2^* = 2x_3^* = \frac{2c_1}{3}(1 - \exp(-\sigma_1)) = 2.1071$. We have plotted the convergence of source rates with iteration time for all flows in Fig. 13.

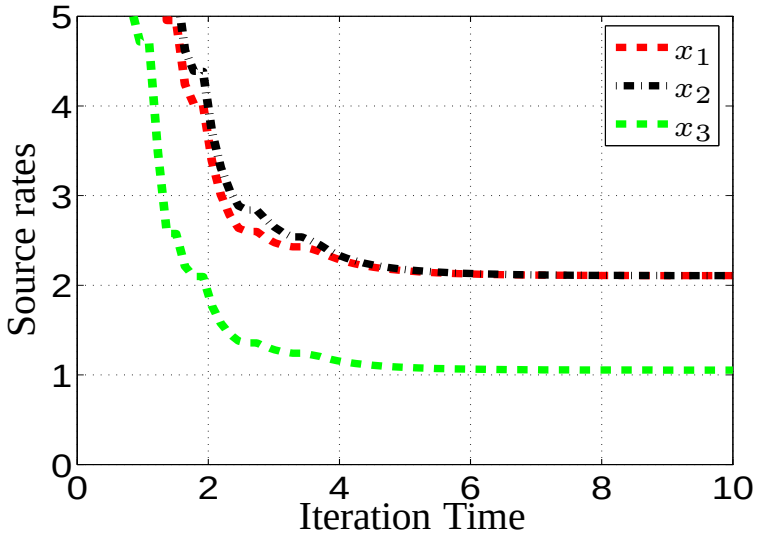

Fig. 13. Distributed resource allocation for three-flow two-hop network.

F. NS-2 Experiments

We implemented our transport layer protocol on Network Simulator (NS-2). Unlike TCP, this transport layer protocol is rate based. The various implementation details are given in the following sections, followed by results and analysis.

1. Source Dynamics

Each Packet header in the source contains the following additional fields:

1.Dissatisfaction of the source. Dissatisfaction is calculated at a per packet basis. It is done as per the following computation at the destination node.

$$\dot{w}(t) = 1000\,(\tilde{\sigma}_r - \sigma_r)^+_{w_r}\,. \tag{3.23}$$

$\tilde{\sigma}_r$ is the quality degradation the source sees. σ_r is the quality degradation the source is willing to tolerate. $w(t)$ is the current dissatisfaction for the particular source. Using

the dissatisfaction in the previous time instant the current source dissatisfaction can be calculated.

2.Route Price (q_r), initialized to zero.

$$q_r = \sum p_l \tag{3.24}$$

p_l is the real queue length at link l.

At the destination node, q_r for a particular source is used to set the source rate till the next interval. The source rate is given by the following equation:

$$x_r = \frac{a}{q_r} \tag{3.25}$$

To scale this equation to the rates of the link capacities, we do a modification as given below:

$$x_r = \frac{a}{\frac{q_r}{10^6}} \tag{3.26}$$

a is set to a value of 100, so that convergence is fast.

2. Link (Router) Dynamics

In order to implement the virtual capacity of links, we need to shape the traffic arriving at links. We achive such traffic shaping through a system of two queues – (i) the traffic shaping (TS) queue, and (ii) the router queue.

The Traffic Shaping (TS) Queue: The purpose of this queue is to shape traffic entering the router queue. Since our system requires decoupling of the real load y_l on link l from the actual load $\tilde{y}_l$, we need to either add or subtract packets arriving at the link. For example, if two sources (S_1, S_2) are using link L, then the whenever a packet from S_1 or S_2 arrive they are enqueued into the TS queue rather than into the router

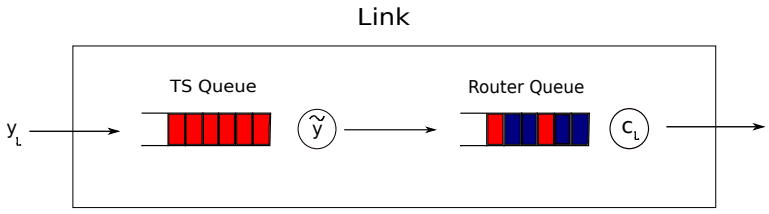

Fig. 14. Link in our system comprises of the TS-queue and the router queue.

queue. The TS queue is drained at rate $\tilde{y}_l$. We implement such a drain rate by means of a token generator at each link. Tokens are created at rate $\tilde{y}_l$. Each token obeserves the TS queue. If there is any packet in the TS queue, it places the packet at the head of the TS queue into the router queue. Otherwise, the token itself is placed in the router queue. Delays are created due to the time that packets spend in the TS queue, but we will see that these delays are small. We choose a large buffer size ($10,000$ packets) to ensure that few packets are lost.

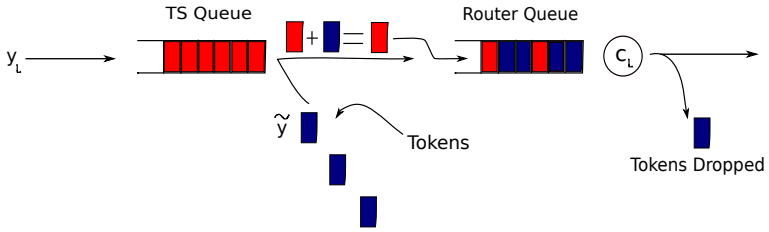

Fig. 15. TS-queue in our system

The Router Queue: This queue is the output queue at the link l. It is drained at rate c_l. The delay of this queue maps to the fucntion $V_l(\tilde{y}_l)$. In other words, the delays seen at this queue are used to calculate end-to-end dissatisfaction at sources. However, the overall packet delays are actually the sum of the TS delay and the router delay. Note that some of the packets on this queue might actually be tokens that were generated

at the TS queue. These tokens are dropped immediately upon reception at the next hop TS queue.

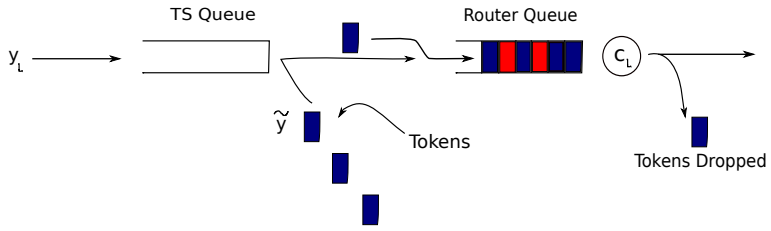

Fig. 16. Router queue in our system

At periodic time intervals the $\tilde{y}$ is updated according to the following equation:

$$\tilde{y} = c_l - \sqrt{\frac{\nu_l}{p_l}} \tag{3.27}$$

where c_l is the link capacity, ν_l is the link dissatisfaction calculated as follows:

$$\nu_l = \sum_{r \in S} w_r \tag{3.28}$$

p_l is the real queue length at the link. As in previous section, we scale ν_l and p_l by a constant so that there is an effect on $\tilde{y}$ when there are changes in either of the quantities. The ν_l is changed as follows:

$$\nu_l = 1000. \sum_{r \in S} w_r \tag{3.29}$$

and the p_l is modified as follows:

$$p_l = \frac{p_l}{10^6} \tag{3.30}$$

The equation for $\tilde{y}_l$ becomes:

$$\tilde{y} = c_l - \sqrt{\frac{10^9 \cdot \nu_l}{p_l}} \tag{3.31}$$

Unlike the analytical proof presented earlier, we note that in reality it takes a while for the impact of changing $\tilde{y}_l$ to be felt on the end-to-end delay. Therefore, we do not change $\tilde{y}_l$ at the same time scale as source rates. Instead, we do so periodically at each link. The times at whcih each link changes its value of $\tilde{y}_l$ are not synchronized. The objective is to get the source rates to converge to $\tilde{y}_l$. The $\tilde{y}_l$ is again changed after some time interval, assuming that the sum of source rates converged to that $\tilde{y}_l$, and hence the observed delay is $V_l(\tilde{y}_l)$.

3. Results

We first simulate the Three-Flows Two-Hop Network as a basic case to study the protocol we have developed. Then we proceed to two realistic topologies presented in [29] to conduct the simulations. Our objective is to study the performance of our value-aware controller in different networking scenarios.

a. Three-Flows Two-Hop Network

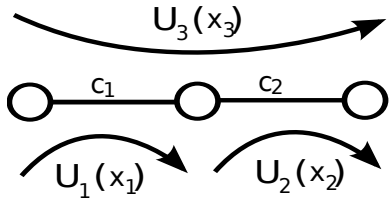

Fig. 17. Three flows sharing a two-link network.

As shown in the above figure, the three flows are x_1, x_2 and x_3. c_1 and c_2 are taken to be 15Mb.

Let T_i denote the propogation delay for flow i with no queueing delays. In the system considered $T_1 = 40$ms, $T_2 = 40$ms and $T_3 = 80$ms. The following two cases are studied here.

1. Flow x_3 with a tight delay constraint
2. All flows with a loose delay constraint.

Case 1: In this case the following parameters for the flows are considered:

$\sigma_1 = 1000$s, $\sigma_2 = 1000$s, $\sigma_3 = 0.085$s. A σ_i value is the allowed dissatisfaction for the flow i. In this case we have set a very high value for flows 1 and 2, while for flow 2 the dissatisfaction allowed is very low and nearly equals the propogation delay of 80ms, and allows a queuing delay of 5ms for the two intermediate queues at router 1 and router 2. The tight delay constraint on flow 3 has an effect on the other two flows which share the link with it. In Figure 18 we have plotted the three rates for flows x_1, x_2 and x_3. Figure 19 shows the acceptable delay for packets for flow 3, the control delay achieved and the actual total delay for packets to reach node 3 from node 1. Figure 20 plots the $\tilde{y}$ at router node 1.

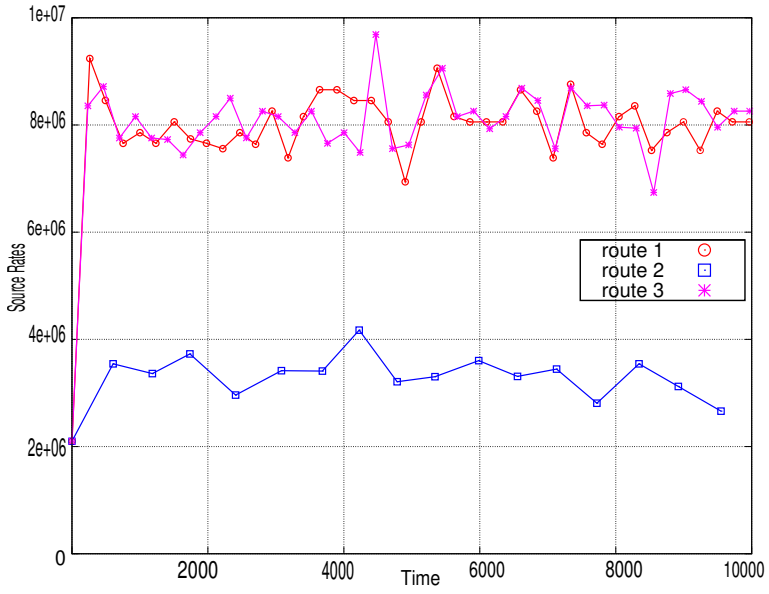

Fig. 18. Rates for the routes 1, 2 and 3 for a realizable but, tight constraint

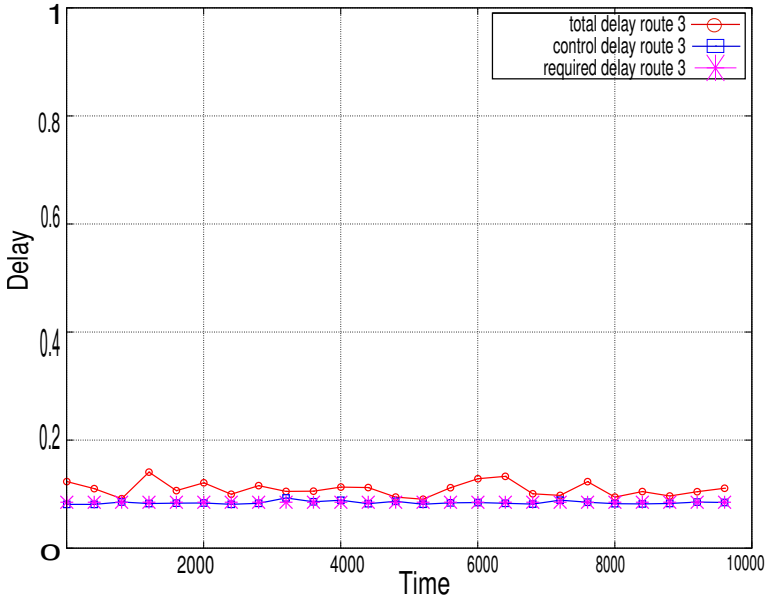

Fig. 19. Delay experienced by route 3, as plotted against the required delay and the control delay

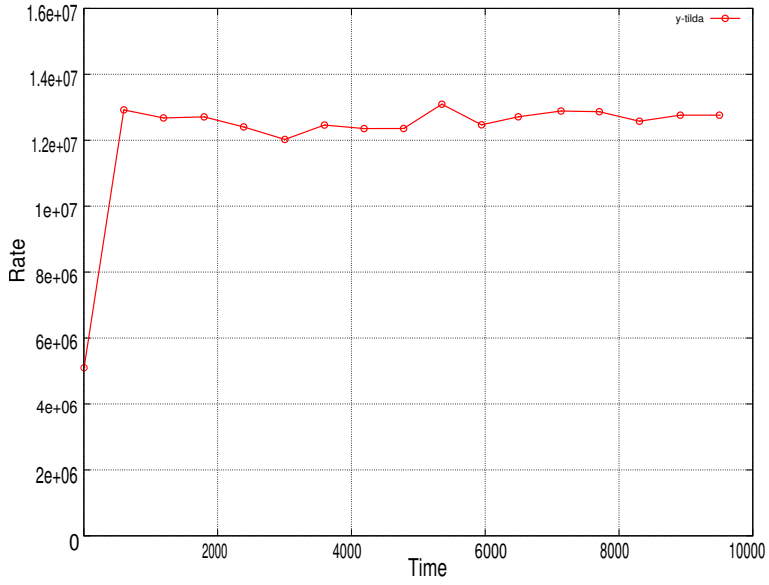

Fig. 20. $\tilde{y}$ at router node 1

Case 2: In this case the following parameters for the flows are considered: $\sigma_1 = 1000$s, $\sigma_2 = 1000$s, $\sigma_3 = 1000$s. In Figure 21 we have plotted the three rates for flows x_1, x_2 and x_3. Figure 22 plots the $\tilde{y}$ at router node 1. As can be seen, $\tilde{y}$ is equal to the link capacity, since the dissatisfaction is very high.

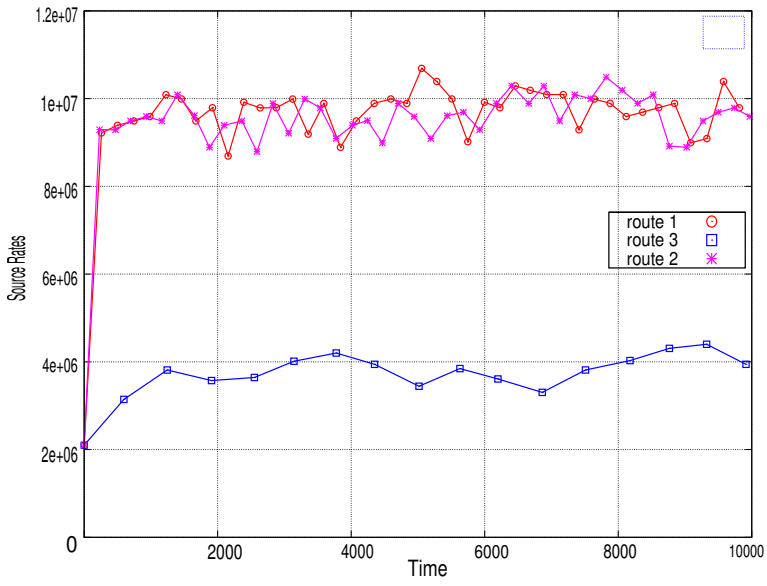

Fig. 21. Rates for the routes 1, 2 and 3 for a realizable loose constraint

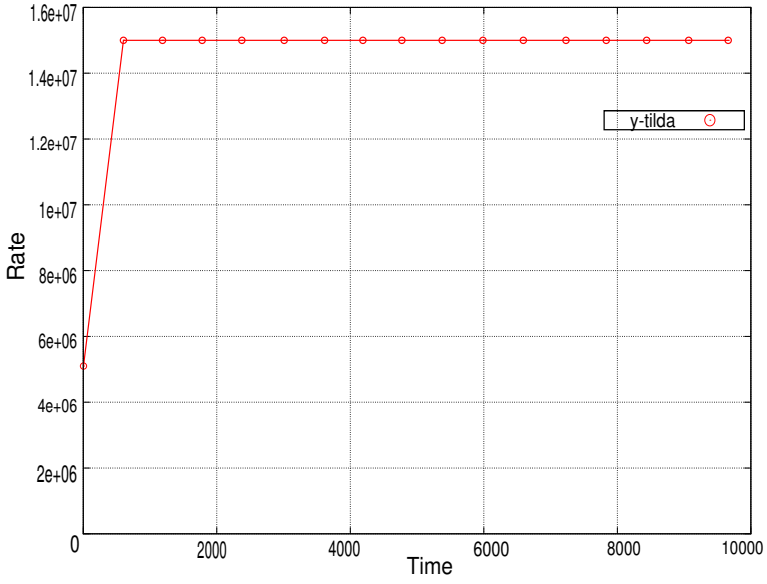

Fig. 22. $\tilde{y}$ at router node 1

b. Abilene Topology

Our first realistic network represents the major nodes of the Abilene network topology [34]. The network consists of high bandwidth links, and connects several universities and research labs. Traffic consists of large scale data transfers (low quality constraints) and distributed computation (where flows have strict delay constraints). Here we consider the same two cases as discussed in the Three-Flows Two-Hop case.

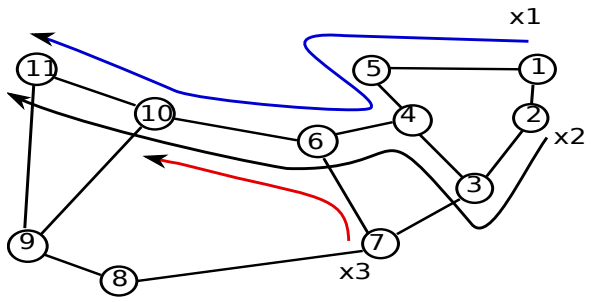

Fig. 23. Abeline topology

As shown in Figure **??**, the three flows are x_1, x_2 and x_3. In the system considered $T_1 = 50$ms, $T_2 = 45$ms and $T_3 = 25$ms. The following two cases are studied here.

1. Flow x_1 with a tight delay constraint
2. All flows with a loose delay constraint.

Case 1: In this case the following parameters for the flows are considered:

$\sigma_1 = 1000$s, $\sigma_2 = 1000$s, $\sigma_3 = 0.055$s. A σ_i value is the allowed dissatisfaction for the flow i. In this case we have set a very high value for flows 2 and 3, while for flow 1 the dissatisfaction allowed is very low and nearly equals the propogation delay of 50ms, and allows a queuing delay of 5ms for the five intermediate queues at routers 1, 5, 4, 6 and 10. The tight delay constraint on flow 1 has an effect on the other two

flows which share the link between routers 6 and 10 with it.

In Figure 24 we have plotted the three rates for flows x_1, x_2 and x_3. Figure 25 shows the acceptable delay for packets for flow 1, the control delay achieved and the actual total delay for packets to reach node 11 from node 1. Figure 26 plots the $\tilde{y}$ at router node 10 which is shared by all the three flows.

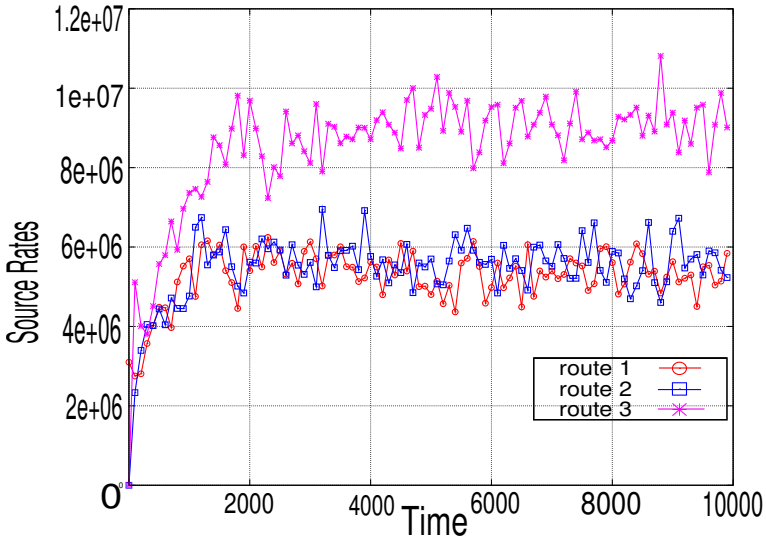

Fig. 24. Rates for the routes 1, 2 and 3 for a realizable but, tight constraint

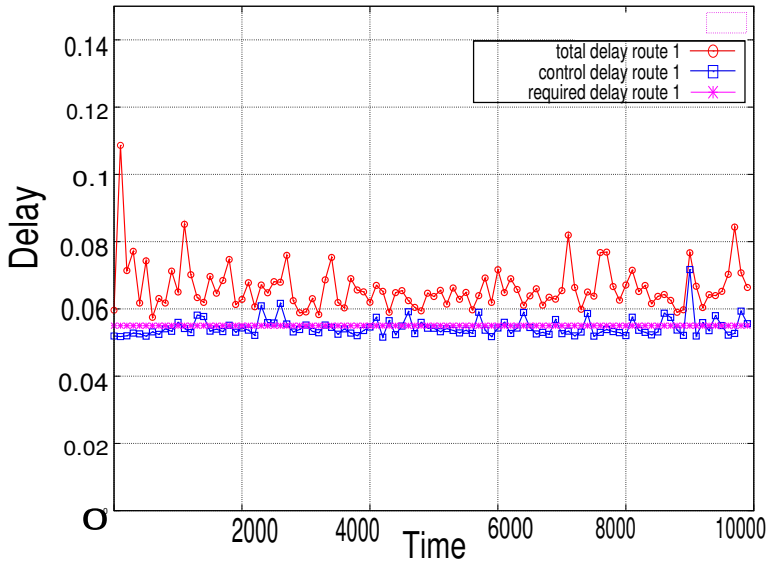

Fig. 25. Delay experienced by route 1, as plotted against the required delay and the control delay

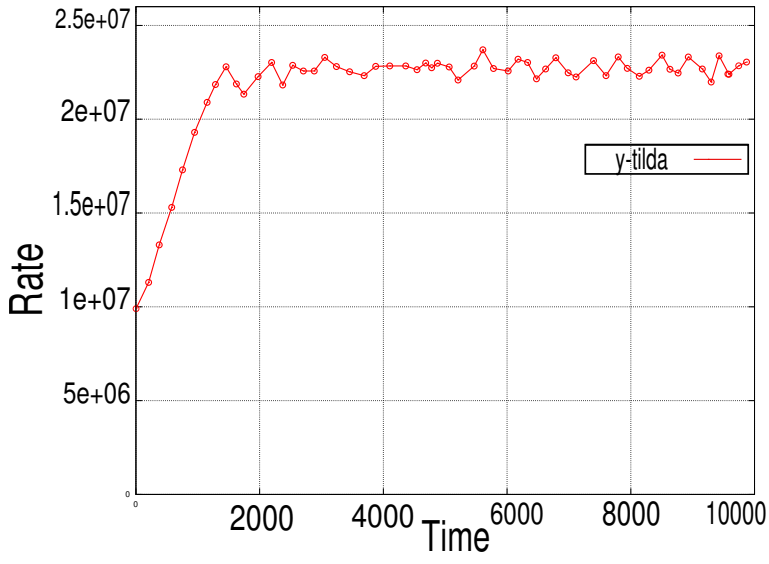

Fig. 26. $\tilde{y}$ at router node 10

Case 2: In this case the following parameters for the flows are considered: $\sigma_1 = 1000$s, $\sigma_2 = 1000$s, $\sigma_3 = 1000$s. In Figure 27 we have plotted the three rates for flows x_1, x_2 and x_3. Figure 28 plots the $\tilde{y}$ at router node 10. As can be seen, $\tilde{y}$ is equal to the link capacity, since the dissatisfaction is very high for all the flows using that link.

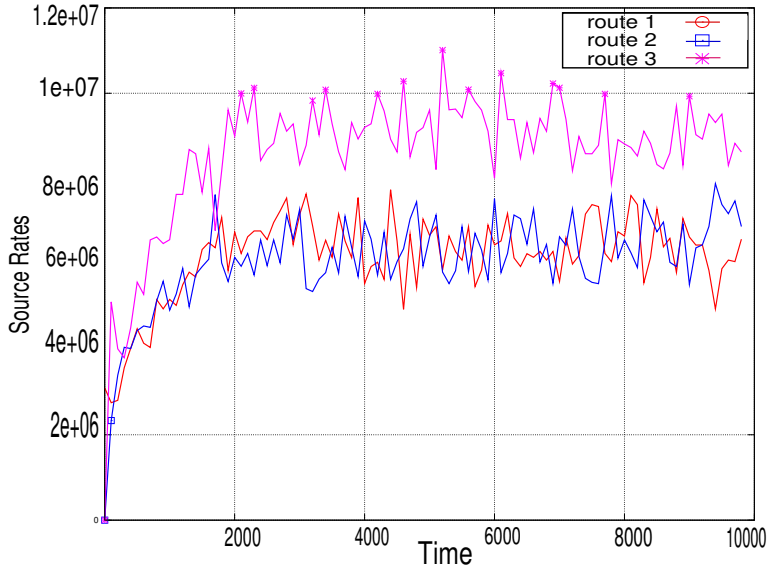

Fig. 27. Rates for the routes 1, 2 and 3 for a realizable loose constraint

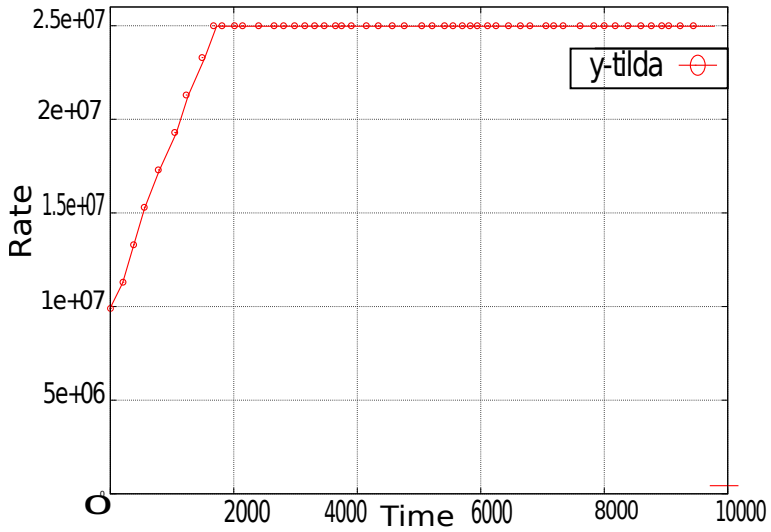

Fig. 28. $\tilde{y}$ at router node 10

c. Abilene Topology: ON-OFF flows

In the *abilene* topology we study the effect of ON-OFF flows on the dynamics of the system. The Figure 23 shows flows $x_1, x_2 and x_3$. x_1 and x_2 flows are present in the system for T time units. The flow x_3 enters the system after t_1 time units and leaves after t_2 time units, where $t_1 < t_2 < T$. The dissatisfaction for $x_1 and x_2$ is very high, while the dissatisfaction for x_3 is very low, i.e. the delay constraint for packets for flow x_3 is very tight. Figure 29 shows how the rates go down as soon as the third flow starts. This is due to the low dissatisfaction of the flow. Similarly Figure 30 shows the corresponding $\tilde{y_l}$ which falls down as soon as the new flow enters, and picks up again as soon as the flow leaves the system.

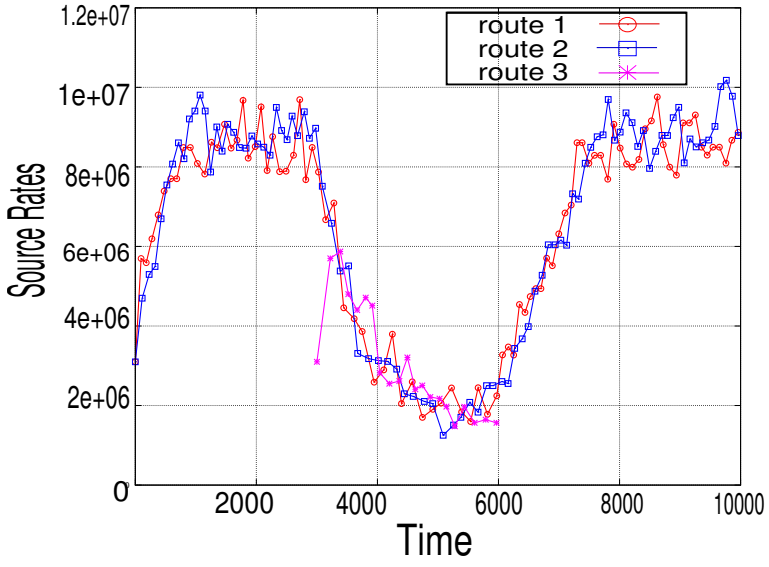

Fig. 29. Rates for flows x_1, x_2, x_3

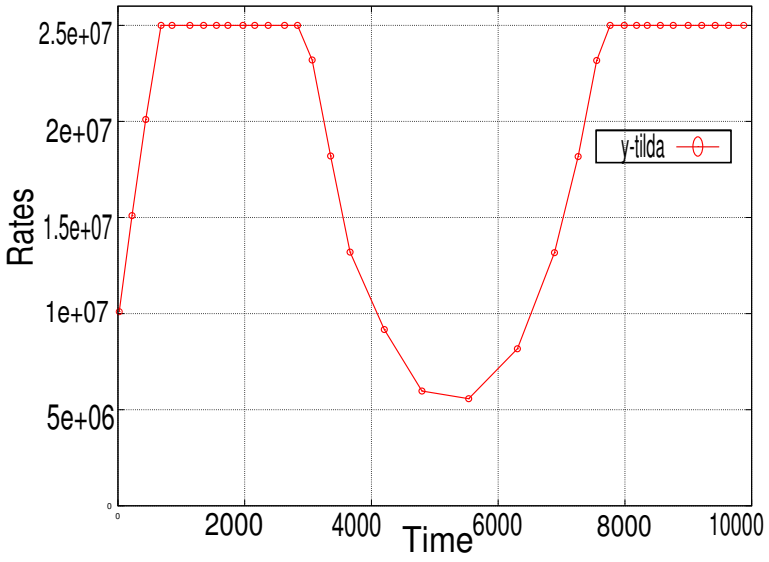

Fig. 30. $\tilde{y}$ at router node 10

d. Access-Core Topology

Our second realistic network is an *access-core* topology. It represents a paradigm similar to commercially available Internet access, wherein users have a relatively small access bandwidth (from homes and businesses), connected together by resource rich core-network. User bandwidth is constrained, either directly at the final hop into the home, or at a neighborhood head-end. Applications such as P2P file transfers (low quality constraint), as well as voice and video calls (higher quality constraints) result in end-to-end traffic on such a topology.

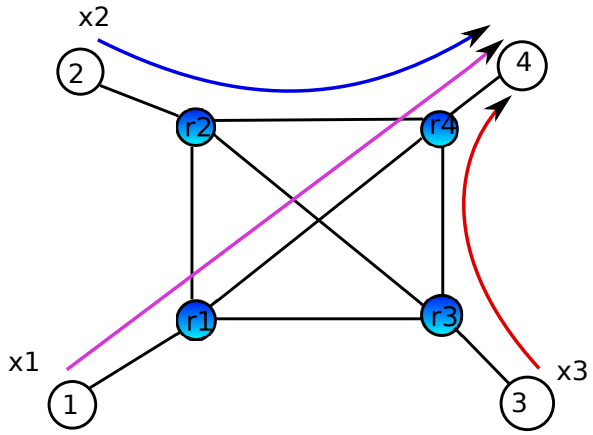

Fig. 31. Access core topology

As shown in Figure 31, the three flows are x_1, x_2 and x_3. In the system considered $T_1 = 20$ms, $T_2 = 25$ms and $T_3 = 20$ms. The following two cases are studied here.

1. Flow x_1 with a tight delay constraint
2. All flows with a loose delay constraint.

Case 1: In this case the following parameters for the flows are considered:

$\sigma_1 = 0.025$s, $\sigma_2 = 1000$s, $\sigma_3 = 1000$s. In this case we have set a very high value for flows 2 and 3, while for flow 1 the dissatisfaction allowed is very low and nearly

equals the propogation delay of 20ms, and allows a queuing delay of 5ms for the three intermediate queues at routers 1, r_1 and r_4. The tight delay constraint on flow 1 has an effect on the other two flows which share the link between r_4 and 4 with it.

In Figure 32 we have plotted the three rates for flows x_1, x_2 and x_3. Figure 33 shows the acceptable delay for packets for flow 1, the control delay achieved and the actual total delay for packets to reach node 4 from node 1. Figure 34 plots the $\tilde{y}$ at router r_4 which is shared by all the three flows.

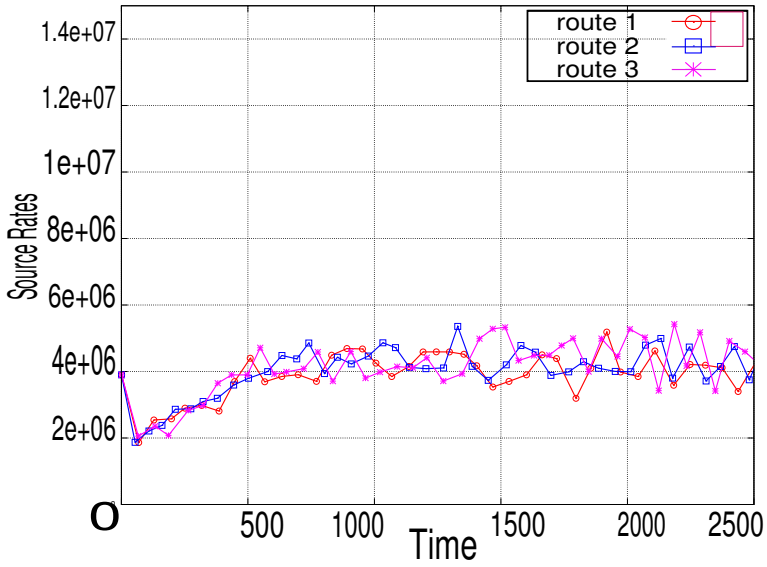

Fig. 32. Rates for the routes 1, 2 and 3 for a realizable but, tight constraint

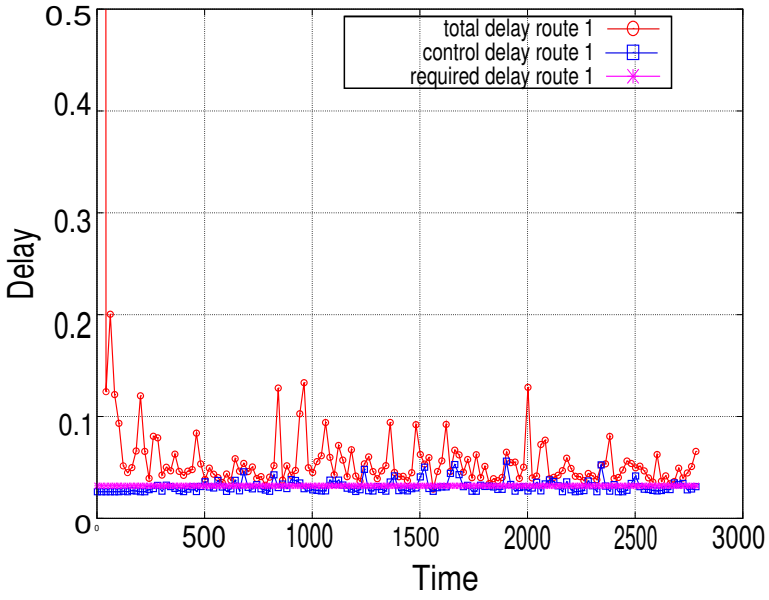

Fig. 33. Delay experienced by route 1, as plotted against the required delay and the control delay

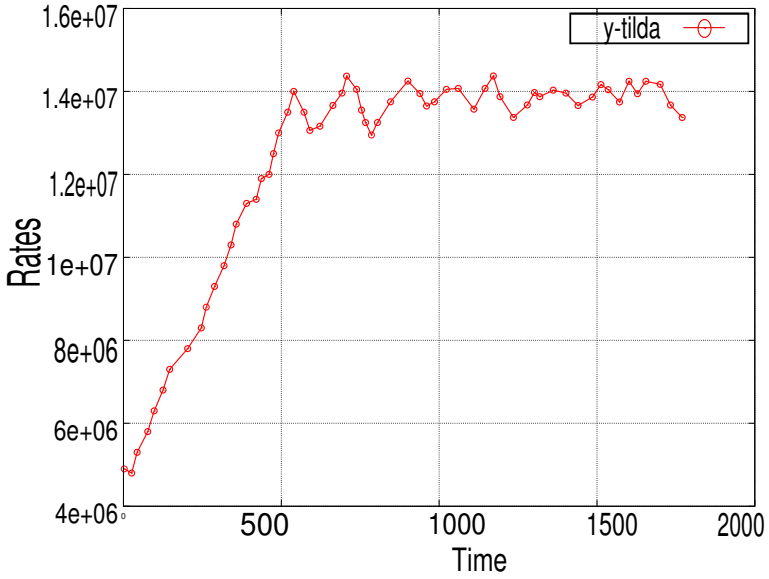

Fig. 34. $\tilde{y}$ at router node 10

Case 2: In this case the following parameters for the flows are considered: $\sigma_1 = 1000$s, $\sigma_2 = 1000$s, $\sigma_3 = 1000$s. In Figure 35 we have plotted the three rates for flows x_1, x_2 and x_3. Figure 36 plots the $\tilde{y}$ at router node r_4. As can be seen, $\tilde{y}$ is equal to the link capacity, since the dissatisfaction is very high for all the flows using that link.

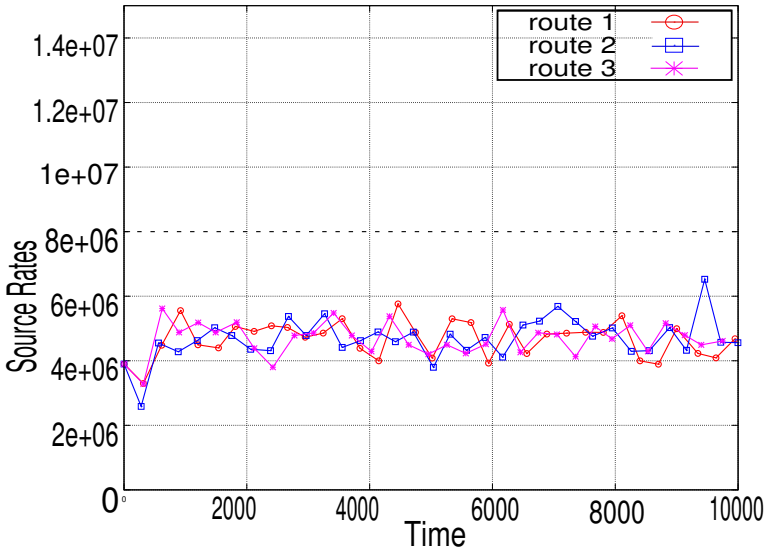

Fig. 35. Rates for the routes 1, 2 and 3 for a realizable loose constraint

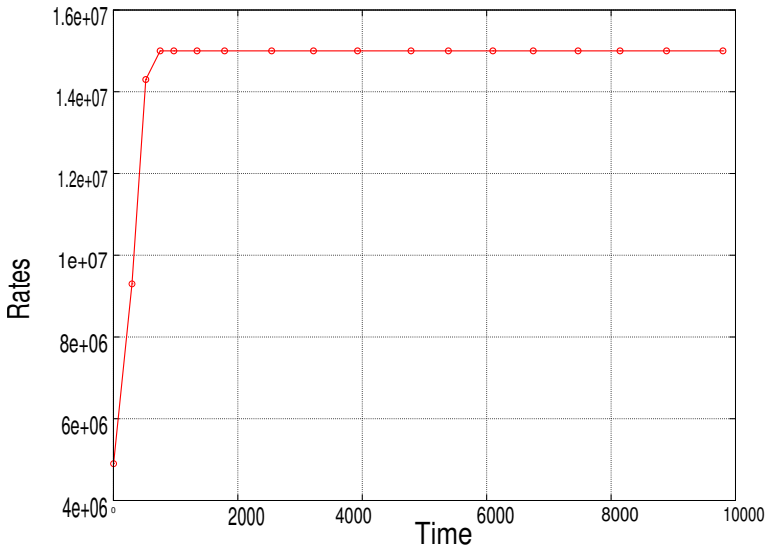

Fig. 36. $\tilde{y}$ at router node 10

CHAPTER IV

CONCLUSIONS

A. Implicit Learning for Explicit Discount Targeting in Online Social Networks

We developed distributed schemes for targeted coupon delivery using online social networks. The objective was to create a two-tier price structure for maximum revenue extraction by selective discounting. We designed systems that allow users to obtain coupons from their neighbors, and incentivize these neighbors by rewarding them for coupon forwarding. We proved how backpressure ideas could be used to achieve a optimal solution, and also how to use it to obtain a simpler (albeit less efficient) scheme. Future work includes dealing with potential malicious users, and a testbed implementation.

B. Value-aware Resource Allocation for Service Guarantees in Networks

In this work we considered the design of a distributed resource allocation algorithm that would allow each individual flow to specify its measure of value. We assumed that every flow passing through a link suffers a certain quality-degradation due to the load on the link, and that such degradation adds up over the multiple links that the flow traverses. The objective is to ensure that the system throughput is maximized in a fair manner, subject to each flow's quality of service satisfying a hard constraint. Our aim was to ensure that the algorithm should be simple, use local information, and the relays need not maintain per-flow information.

We first showed that attempting to solve this problem by the usual optimization decomposition techniques in Primal formulation yield approximate solutions, and in Dual formulation centralized solutions. However, the observation that decoupling

the link-load from the quality degradation using a secondary variable that we call *effective-capacity*, allows us to design such a controller. Under our scheme, the source chooses its rate based on a *route price*, and it declares a *dissatisfaction* based on the quality of service that it sees. Links choose an effective-capacity based on dissatisfaction and *link-price*, and modify the price as if the effective-capacity were the actual capacity of the link. The control scheme only requires that links be aware of *aggregate* quantities of the flows using them, and the sources perform computations solely based on the parameters obtained from the links they traverse, hence satisfying our requirements.

We studied various topologies, and used ns-2 simulations to show the performance acheivable by this protocol. We observed that if a flow declares a very low dissatisfaction (achievable), it will be able to achieve it. In order to do so, the *effective capacity* would go down and the flows using the common links with this flow will need to cut down on their rates. On the other hand if the dissatisfaction declared by all the flows is high, all flows will use up the entire link capacity, since the *effective capacity* of the link would be equal to the link capacity. As mentioned in the implementation section of the ns-2 simulations, this protocol can be easily implemented in the current routers and end hosts.

REFERENCES

[1] Facebook, 2009, http://www.facebook.com/.

[2] Orkut, 2009, http://www.orkut.com/.

[3] mGinger, 2009, http://www.mginger.com/.

[4] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, December 1992.

[5] M. Neely, E. Modiano and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proceedings IEEE Infocom.*, Miami, FL, 2005.

[6] A. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Systems*, vol. 50, no. 4, pp. 401–457, August 2005.

[7] A. Eryilmaz and R. Srikant, "Joint congestion control, routing and MAC for stability and fairness in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514–1524, August 2006.

[8] L. Chen, S. H. Low, M. Chiang and J. C. Doyle, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," *IEEE Infocom*, Barcelona, Spain, April 2006.

[9] F. P. Kelly, "Multi-armed bandits with discount factor near one: The Bernoulli Case," *Annals of Statistics*, vol. 9, pp. 987–1001, 1982.

[10] P. Auer, N. Cesa-Bianchi, Y. Freund and R.E. Schapire, "The Nonstochastic Multiarmed Bandit Problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2003.

[11] N. Abe, A.W. Biermann and P.M. Long, "Reinforcement learning with immediate rewards and linear hypotheses," *Algorithmica*, vol. 37, no. 4, pp. 263–293, 2003.

[12] X. Lin and N. Shroff, "Joint rate control and scheduling in multihop wireless networks," *IEEE Conference on Decision and Control*, Paradise Island, Bahamas, 2004.

[13] L. Bui, R. Srikant and A. L. Stolyar, "Optimal resource allocation for multicast flows in multihop wireless networks," *Philosophical Transactions of the Royal Society*, vol. 35, issue 3, pp. 43–43, 2008.

[14] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.

[15] F. P. Kelly, "Models for a self-managed Internet," *Philosophical Transactions of the Royal Society*, A358, pp. 2335–2348, 2000.

[16] F. P. Kelly,"Mathematical modeling of the Internet," *in Mathematics Unlimited - 2001 and Beyond (Editors B. Engquist and W. Schmid)*, Berlin, Springer-Verlag, 2001, pp. 685–702.

[17] R. Srikant, "The Mathematics of Internet Congestion Control," Birkhauser, 2004.

[18] S. Shakkottai and R. Srikant, "Network optimization and control," *Foundations and Trends in Networking, Now Publishers*, vol. 2, 2008.

[19] M. Chiang, S. H. Low, A. R. Calderbank and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *in Proceedings of IEEE*, vol. 95, no. 1, pp. 255-213, 2007.

[20] R. Stevens, *TCP/IP Illustrated, Volume 1*, Addison-Wesley, 1994.

[21] L. L. Peterson and B.S. Davie, *Computer Networks: A Systems Approach*, Morgan-Kaufman, 1999.

[22] F. P. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.

[23] F. P. Kelly, A. Maulloo and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[24] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, pp. 861–875, December, 1999.

[25] G. Vinnicombe, "On the stability of networks operating TCP-like congestion control," *Proceedings of the IFAC World Congress*, Barcelona, Spain, 2002.

[26] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 83–91, April, 2003.

[27] D. X. Wei, C. Jin, S. H. Low and S. Hegde, "FAST TCP: motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246–1259, December, 2006.

[28] S. Liu, T. Başar and R. Srikant, "TCP-Illinois: a loss and delay-based congestion control algorithm for high-speed networks," *in valuetools '06: Proceedings of the 1st International Conference on Performance Evaluation Methodologies and tools*, Pisa, Italy, October, 2006.

[29] J. He, M. Suchara and J. Rexford and M. Chiang, "Rethinking Internet Traffic Management: From Multiple Decompositions to A Practical Protocol," in *Proc. CoNEXT*, New York, NY, December, 2007.

[30] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "*An Architecture for Differentiated Services*," June, 1997, RFC 2475 (http://tools.ietf.org/html/rfc2475.txt).

[31] H. Khalil, *Nonlinear Systems*, 2nd edition, Prentice Hall, Upper Saddle River, NJ, 1996.

[32] Lingjia Liu, Parimal Parag, Jia Tang, Wei-Yu Chen and Jean-François Chamberland, "Resource allocation and quality of service evaluation for wireless communication systems using fluid models," *IEEE Transactions on Information Theory*, vol. 53, no. 5, pp. 1767–1777, May, 2007.

[33] W. Kang, F. P. Kelly, N. H. Lee and R. J. Williams, ' "State space collapse and diffusion approximation for a network operating under a fair bandwidth sharing policy," to appear in *Annals of Applied Probability*.

[34] Internet2, 2009, available at `http://www.internet2.edu/network/`.

APPENDIX A

PROOF OF LEMMA 1

First, it is easy to verify that the optimal solution satisfies $y^i_{(j,s)} \leq x^{il}_j$ because otherwise, the store can extract more profit by reducing the number of coupons sent to user j. Based on that, OPT 1 can be re-written as:

$$\begin{aligned}
\textbf{OPT1}: \qquad & \max \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{B}_i} \left(q^i y^i_{(j,s)} + p^i \hat{x}^{ih}_j\right) \\
s.t. \quad & \sum_{i\in\mathcal{S}} \sum_{j:(m,j)\in\mathcal{L}, j\neq s_i} y^i_{(m,j)} \leq \mu_m, \forall m \in \mathcal{N} \\
& \sum_{j:(m,j)\in\mathcal{L}} y^i_{(m,j)} = \sum_{n:(n,m)\in\mathcal{L}} y^i_{(n,m)} \forall m \in \mathcal{N} \\
& y^i_{(j,s)} \leq x^{il}_j \\
& y^i_{(m,j)} = 0 \text{ if } m \in \mathcal{B}_i \text{ and } j \neq s
\end{aligned}$$

Since $\hat{x}^{ih}_j$ are constants, the objective is equivalent to

$$\max \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{B}_i} \left(q^i y^i_{(j,s)}\right) = \max \sum_{i\in\mathcal{S}} q^i \left(\sum_{j\in\mathcal{B}_i} y^i_{(j,s)}\right).$$

Thus, the lemma holds.

APPENDIX B

PROOF OF THEOREM 2

The analysis follows the Lyapunov drift used in [5–7].

Define $\mathbf{Q}[t] = \{Q_j^i[t], \tilde{Q}_j^i[t]\}_{j\in\mathcal{N}, i\in\mathcal{S}}$. It is easy to verify that $\mathbf{Q}[t]$ is a Markov chain. Further, given $\gamma_j^i \leq b_j^i$ for all j and i, we can obtain that for any $j \in \mathcal{B}_i$, the following holds

$$\begin{aligned} & Q_j^i[t+1] + \tilde{Q}_j^i[t+1] \\ = & \left(Q_j^i[t] + \tilde{Q}_j^i[t] + \sum_{m:(m,j)\in\mathcal{L}} y_{(m,j)}^i[t] - \gamma_j^i \right)^+ . \end{aligned}$$

Next, consider a Lyapunov function such that

$$V[t] = \frac{1}{2} \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{B}_i} \left(Q_j^i[t] + \tilde{Q}_j^i[t] \right)^2 .$$

Following the analysis in [7], it can be shown that there exists $B > 0$, independent of $\mathbf{Q}[t]$, such that

$$\begin{aligned} & \mathbf{E}\left[V[t+1] - V[t] | \mathbf{Q}[t] \right] \\ \leq & \; B + \sum_{i\in\mathcal{S}} Q_d^i[t] \mathbf{E}\left[\Theta^i[t] - \sum_{j:(d,j)\in\mathcal{L}} y_{(d,j)}^i[t] \middle| \mathbf{Q}[t] \right] \\ & + \sum_{i\in\mathcal{S}, j\neq d} \left(Q_j^i[t] + \tilde{Q}_j^i[t] \right) \times \\ & \mathbf{E}\left[\left(\sum_{m:(m,j)\in\mathcal{L}} y_{(m,j)}^i[t] - \sum_{n:(j,n)\in\mathcal{L}} y_{(n,j)}^i[t] \right) \middle| \mathbf{Q}[t] \right] . \end{aligned}$$

Letting $\check{\Theta}^i = \sum_{j:(d,j)\in\mathcal{L}} \check{y}_{(j,d)}^i$, where $\check{y}_{(j,d)}^i$ is the optimal solution to OPT 2, we further

obtain that

$$\begin{aligned}
& \mathbf{E}\left[V[t+1]-V[t]|\mathbf{Q}[t]\right] \\
= \; & B_1+\sum_{i\in\mathcal{S}}\left(Q_d^i[t]-Q_T q^i\right)\mathbf{E}\left[\Theta^i[t]-\check{\Theta}^i\middle|\mathbf{Q}[t]\right] \\
& +\sum_{i\in\mathcal{S}}Q_d^i[t]\check{\Theta}^i \\
& -\sum_j\sum_{m:(j,m)\in\mathcal{L}}\sum_{i\in\mathcal{S}}y_{(j,m)}^i[t]\times \\
& \quad\left(Q_j^i[t]+\tilde{Q}_j^i[t]-Q_m^i[t]-\tilde{Q}_m^i[t]\right)
\end{aligned}$$

Next we have that

$$\begin{aligned}
& \sum_{(j,m)\in\mathcal{L}}\sum_{i\in\mathcal{S}}y_{(j,m)}^i[t]\left(Q_j^i[t]+\tilde{Q}_j^i[t]-Q_m^i[t]-\tilde{Q}_m^i[t]\right) \\
\geq_{(a)} & \sum_{(j,m)\in\mathcal{L}}\sum_{i\in\mathcal{S}}\check{y}_{(j,m)}^i\left(Q_j^i[t]+\tilde{Q}_j^i[t]-Q_m^i[t]-\tilde{Q}_m^i[t]\right) \\
\geq & \sum_{i\in\mathcal{S}}Q_d^i[t]\check{\Theta}^i,
\end{aligned}$$

where inequality (a) holds due to backpressure routing, and

$$\left(Q_{d_i}^i[t]-Q_T q^i\right)\left(\Theta^i[t]-\check{\Theta}^i\right)\leq 0$$

holds according to the definition of the rate controller. Thus, according to the Foster's criterion, we can conclude that $\mathbf{Q}[t]$ is positive recurrent, which implies that $\lim_{t\to\infty}\mathbf{E}[V[t]]<\infty$.

Since $\mathbf{Q}[t]$ is positive recurrent, we further have

$$\begin{aligned} & \frac{1}{T}\left(\mathbf{E}\left[V[T]\right]-\mathbf{E}\left[V[0]\right]\right) \\ = & \frac{1}{T}\sum_{t=1}^{T}\left(\mathbf{E}\left[\mathbf{E}[V[t]|\mathbf{Q}[t]]\right]-\mathbf{E}\left[\mathbf{E}[V[t-1]|\mathbf{Q}[t-1]]\right)\right. \\ \leq & B_1+\sum_i Q_T q^i\left(\frac{\sum_{t=0}^{T}\Theta^i[t]}{T}-\check{\Theta}^i\right), \end{aligned}$$

which implies that

$$\sum_i q^i\left(\check{\Theta}^i-\frac{\sum_{t=0}^{T}\Theta^i[t]}{T}\right)\leq\frac{B}{Q_T}+\frac{\mathbf{E}\left[V[T]-V[0]\right]}{T}.$$

Note that

$$\sum_i q^i\left(\check{\Theta}^i-\frac{\sum_{t=0}^{T}\Theta^i[t]}{T}\right)\geq 0$$

because the network is stable and the $\check{\Theta}^i$ is the optimal solution to OPT 2. Thus, we have that

$$0\leq\sum_i q^i\left(\check{\Theta}^i-\lim_{T\to\infty}\frac{\sum_{t=0}^{T}\Theta^i[t]}{T}\right)\leq\frac{B}{Q_T},$$

which leads to equality (2.11). Furthermore, since the queues are stable, so when $T\to\infty$, almost all coupons are consumed, which implies that equality (2.12).

APPENDIX C

PROOF OF THEOREM 3

We use a Lyapunov argument, with the Lyapunov function

$$J[k] = \left(\gamma_j^i[k] - \hat{\gamma}_j^i\right)^2.$$

Then we have

$$\begin{aligned} J[k+1] &- J[k] \\ &= \left(\gamma_j^i[k+1]\right)^2 + \left(\hat{\gamma}_j^i\right)^2 - 2\hat{\gamma}_j^i\gamma_j^i[k+1] \\ &\quad -\left(\gamma_j^i[k]\right)^2 - \left(\hat{\gamma}_j^i\right)^2 + 2\hat{\gamma}_j^i\gamma_j^i[k] \\ &= \left(\gamma_j^i[k+1] - \gamma_j^i[k]\right)\left(\gamma_j^i[k+1] + \gamma_j^i[k] - 2\hat{\gamma}_j^i\right) \end{aligned}$$

We have two cases.

Case I: If $\Delta\gamma_j^i\Delta z_j^i[k] > 0$, i.e., $\gamma_j^i[k] < \hat{\gamma}_j^i$, we have from (2.13)

$$J[k+1] - J[k] = \delta(2(\gamma_j^i[k] - \hat{\gamma}) + \delta),$$

which is non-positive except in $\gamma_j^i[k] - \hat{\gamma}_j^i \in [-\delta/2, 0]$.

Case II: $\Delta\gamma_j^i\Delta z_j^i[k] = 0$, i.e., $\gamma_j^i[k] \geq \hat{\gamma}_j^i$, we have from (2.13)

$$J[k+1] - J[k] = -\delta(2(\gamma_j^i[k] - \hat{\gamma}) + \delta),$$

which is non-positive except in $\gamma_j^i[k] - \hat{\gamma}_j^i \in [0, \delta/2]$.

Thus, the system is globally asymptotically stable and $\gamma_j^i - \hat{\gamma}_j^i$ will converge to the interval $[-\delta/2, +\delta/2]$.

VITA

Sankalp Sah, is a graduate student in the Department of Electrical & Computer Engineering at Texas A&M University. He received a degree in, B.Tech Electronics & Communication Engineering from Indian Institute of Technology, Guwahati, India in 2005 and a M.S in Computer Engineering at Texas A&M University in 2010. His interests are in communication networks, specifically in the transport layer protocol and in packet forwarding algorithms.

The typist for this thesis was Sankalp Sah.

Department of Electrical and Computer Engineering

Texas A&M University

214 Zachry Engineering Center

TAMU 3128

College Station. TX 77843-3 128

c/o Srinivas Shakkottai

email : sankalpsah@gmail.com